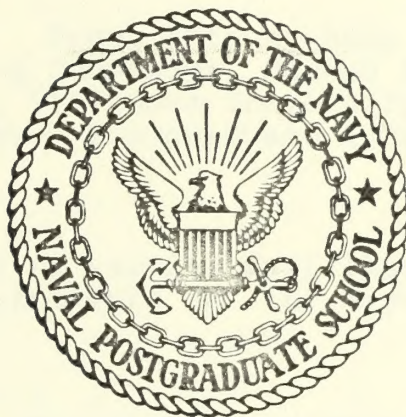


A DIGITAL COMPUTER STUDY OF THE BUCKLING
OF ACTUAL IMPERFECT CYLINDERS -
A MODIFICATION
OF THE COMPUTER PROGRAM SATANS -
THEORY AND USER'S MANUAL
FOR SATANS-I AND SATANS II

Bruce Anthony Ryan

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

A DIGITAL COMPUTER STUDY OF THE BUCKLING
OF
ACTUAL IMPERFECT CYLINDERS -
A MODIFICATION
OF THE COMPUTER PROGRAM SATANS -
THEORY AND USER'S MANUAL
FOR SATANS-I AND SATANS-II

by

Bruce Anthony Ryan

Thesis Advisor:

R. E. BALL

December 1972

Approved for public release; distribution unlimited.

T152259

LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940 /

A Digital Computer Study of the Buckling
of
Actual Imperfect Cylinders -
A Modification of the Computer Program SATANS -
Theory and User's Manual for SATANS-I and SATANS-II

by

Bruce Anthony Ryan
Lieutenant Commander, United States Navy
B.S., Villanova University, 1964
M.S., Naval Postgraduate School, 1972

Submitted in partial fulfillment of the
requirements for the degree of

AERONAUTICAL ENGINEER

from the

NAVAL POSTGRADUATE SCHOOL
December 1972

ABSTRACT

A digital computer program for the geometrically nonlinear static and dynamic response of arbitrarily loaded shells of revolution known as SATANS is modified, removing a requirement that the applied loads be symmetric about a datum meridional plane. The ability to include arbitrary naturally-occurring initial geometric imperfections of any shape, location or extent is added to the program. Two specific axially loaded circular cylinders are analyzed using actual initial imperfections. Detailed numerical studies are conducted to determine the buckling loads of the perfect and imperfect cylinders. Several sets of boundary conditions are considered. An analysis procedure for imperfect shells is proposed. The results indicate that a means to conduct a numerical analysis of an arbitrarily imperfect shell of revolution has been developed.

A study of the buckling behavior of the clamped shallow spherical cap modelled under various boundary conditions at the pole is included as an addendum.

TABLE OF CONTENTS

I.	INTRODUCTION-----	13
II.	DESCRIPTION OF SATANS-----	16
	A. GEOMETRY AND COORDINATE SYSTEM-----	16
	B. METHOD OF SOLUTION-----	17
	C. USER PREPARED SUBROUTINES-----	20
III.	CONSTRUCTION OF SATANS-I-----	23
	A. PACKAGING-----	23
	B. PLOTTING ROUTINE-----	23
	C. MISCELLANEOUS ALTERATIONS-----	25
	D. USE OF SATANS-I-----	27
IV.	CONSTRUCTION OF SATANS-II-----	28
	A. INTRODUCTION OF FULL TRIGONOMETRIC SERIES EXPANSIONS-----	28
	B. INTRODUCTION OF INITIAL IMPERFECTIONS-----	32
	C. COMBINATION OF FULL TRIGONOMETRIC SERIES EXPANSIONS AND INITIAL IMPERFECTIONS-----	36
	D. SUBROUTINE MODIFICATION-----	37
	E. SATANS-II-----	40
	F. USE OF SATANS-II-----	42
	G. VALIDITY OF SATANS-II-----	42
V.	IMPERFECT SHELL ANALYSIS-----	46
	A. PREPARATION OF IMPERFECTION DATA-----	46
	1. Data Generation By Mathematical Description-----	46
	2. Data Generation by Graphical Plots-----	51
	3. Data Generation by Experimental Measurement-----	51

4. Selection of Important Harmonics-----	55
B. BOUNDARY CONDITIONS-----	56
C. ANALYSIS-----	58
1. Buckling of the Thin Cylinder-----	58
2. Introduction of Imperfections-----	70
3. Results-----	79
D. DISCUSSION-----	84
E. PROPOSED ANALYSIS PROCEDURE-----	85
VI. CONCLUSIONS-----	88
A. SATANS-I-----	88
B. SATANS-II-----	88
C. IMPERFECT ANALYSIS-----	89

ADDENDUM

A DIGITAL COMPUTER STUDY OF THE EFFECTS OF VARIOUS ARTIFICIAL BOUNDARY CONDITIONS AT THE POLE OF A SHALLOW SPHERICAL SHELL

I. INTRODUCTION-----	90
II. STATEMENT OF THE PROBLEM-----	92
A. GEOMETRY AND LOADING-----	92
B. POLE CONDITIONS-----	94
C. TYPES OF ANALYSIS-----	97
III. DISCUSSION-----	98
A. THE $\lambda=6$ SHALLOW CAP-----	98
1. Axisymmetric Analysis-----	98
2. Bifurcation Buckling Analysis-----	102
3. Nonlinear Nonsymmetric Collapse Analysis--	105
B. THE $\lambda=18$ SHALLOW CAP-----	108

C. COMPARISONS WITH OTHER RESEARCHERS-----	108
IV. CONCLUSIONS-----	110
APPENDIX A Nonlinear Terms-----	112
APPENDIX B Eigenvalue Problem Matrix Element Definitions-----	115
APPENDIX C Input Data Guide for SATANS-I and SATANS-II-----	117
APPENDIX D Listing of SATANS-I User Prepared Subroutines-----	124
APPENDIX E Listing of SATANS-I-----	128
APPENDIX F Listing of Modified Subroutines and COMMON Statements for SATANS-II-----	210
APPENDIX G Listing of Imperfection Data Handling Program-----	252
LIST OF REFERENCES-----	263
INITIAL DISTRIBUTION LIST-----	265
FORM DD 1473-----	267

LIST OF TABLES

I.	Harmonic, Mode and Fourier Index Correspondence--	38
II.	Fourier Coefficients for Check Loadings-----	44
III.	Imperfect Shell Overall Characteristics-----	52
IV.	Boundary Condition Specifications-----	59
V.	Critical Buckling Loads of Individual Harmonics--	75
VI.	Relative Imperfection Rotation Coefficient Sizes, Shell A-14-----	76
VII.	Harmonic Selection, Shell A-14-----	77
VIII.	Buckling Loads, Minute Versus Full Imperfections-	80
IX.	Load Steps Versus Iterations-----	83
X.	Predicted Buckling Loads of the Imperfect Shell--	86
XI.	Shallow Cap ($\lambda=6$) Buckling Loads and Displacements at a Point-----	99
XII.	Shallow Cap ($\lambda=6$) Buckling Loads Comparison-----	109

LIST OF FIGURES

1.	Shell Geometry and Coordinate System-----	18
2.	Positive Directions for Variables-----	19
3.	Nonlinear Behavior Affecting Convergence-----	21
4.	Premature Program Termination-----	26
5.	SATANS-II Check Loadings-----	43
6.	Simplified Imperfection Data Generation Scheme-----	47
7.	Mathematical Imperfection Generation-----	49
8.	Meridional Imperfection Rotation Representations---	50
9.	Circumferential Trace Number and Meridional Station Number Correspondence-----	53
10.	Data Point Extrapolation Technique for \bar{W} -----	54
11.	Cylindrical Shell Testing Configuration-----	57
12.	Membrane Prebuckle Deflection Shape, Shell A-14----	62
13.	Critical Buckling Modes from Eigenvalue Analysis, Shell A-14-----	64
14.	Buckling Load vs Stations, Membrane Prebuckle Analysis, Shell A-14-----	66
15.	Buckling Load vs Stations, Consistent Analysis, Shell A-14-----	67
16.	Buckling Load vs Buckling Mode, Membrane Prebuckle Analysis, Shell A-14-----	68
17.	Buckling Load vs Buckling Mode, Consistent Analysis, Shell A-14-----	69
18.	Circumferential Imperfection Rotation Representation, ϕ_θ vs θ , Plate 1-----	71
	Plate 2-----	72
19.	Critical Buckling Load vs Imperfection Level-----	82
20.	$\lambda=6$ Shallow Spherical Cap Geometry and Loading-----	93

21. Normal Displacement Error, Axisymmetric Analysis, $\lambda=6$ Cap-----	100
22. Meridional Force Error, Axisymmetric Analysis, $\lambda=6$ Cap-----	101
23. Normal Displacement Error, Bifurcation Buckling Analysis, $\lambda=6$ Cap-----	103
24. Meridional Force Error, Bifurcation Buckling Analysis, $\lambda=6$ Cap-----	104
25. Normal Displacement Error, Nonsymmetric Collapse Analysis, $\lambda=6$ Cap-----	106
26. Meridional Force Error, Nonsymmetric Collapse Analysis, $\lambda=6$ Cap-----	107

TABLE OF SYMBOLS

a	= reference length
b	= nondimensional inplane stiffness
E	= Young's modulus
E_0	= reference Young's modulus
h	= thickness
h_0	= reference thickness
H	= rise of a shallow spherical cap
m	= harmonic number, meridional
$M_s, M_\theta, M_{s\theta}$	= bending and twisting moments per unit length
$m_s, m_\theta, m_{s\theta}$	= nondimensional bending and twisting moment series coefficients
n	= harmonic number, circumferential
$N_s, N_\theta, N_{s\theta}$	= membrane forces per unit length
p	= nondimensional pressure load series coefficient
\hat{Q}_s	= effective transverse force
q, q_s, q_θ	= normal, meridional and circumferential components of applied pressure loading
q/q_0	= ratio of applied load to critical load
R_s, R_θ	= principal radii of curvature
r	= normal distance from the axis of the shell
s	= meridional shell coordinate
$t_s, t_\theta, t_{s\theta}$	= nondimensional membrane force series
t_T	= nondimensional thermal membrane force series coefficient
U, V	= displacements tangent to the meridian and tangent to the parallel circle

u, v	= nondimensional series coefficients of U, V
W	= displacement normal to the reference surface
\bar{W}	= displacement normal to the reference surface due to initial imperfections
w	= nondimensional series coefficient of W
$\beta, \beta_s, \beta_\theta, \beta_{s\theta}$	= nondimensional nonlinear term series coefficients in the strain-displacement equations
Δ	= distance between stations
$\epsilon_s, \epsilon_\theta, \epsilon_{s\theta}$	= nondimensional reference surface strain coefficients
$\bar{\epsilon}_s, \bar{\epsilon}_\theta, \bar{\epsilon}_{s\theta}$	= nondimensional reference surface imperfection strain series coefficients
ζ	= shell coordinate perpendicular to the reference surface
θ	= shell circumferential coordinate
$\eta, \eta_s, \eta_\theta, \eta_{s\theta}$	= nondimensional nonlinear term series coefficients in the equilibrium equations
λ	= nondimensional geometric descriptive parameter for a spherical cap
ν	= Poisson's ratio
σ_0	= reference stress level
ρ	= nondimensional radius
ω_s, ω_θ	= nondimensional curvatures
$\Phi, \Phi_s, \Phi_\theta$	= reference surface rotations
$\bar{\Phi}_s, \bar{\Phi}_\theta$	= reference surface rotations due to initial imperfections
$\phi, \phi_s, \phi_\theta$	= nondimensional series coefficients for reference surface rotations
$\bar{\phi}_s, \bar{\phi}_\theta$	= nondimensional series coefficients for reference surface rotations due to initial imperfections
$[\bar{\Omega}], [\bar{\Lambda}]$	= 4 x 4 boundary condition matrices

$\{l\}$ = 1 x 4 boundary condition matrix

$\underline{E}, \underline{F}, \underline{G}$ = matrices [Refs. 3 and 4]

\sum_1^∞ = a summation, on the harmonic index number n , where $n = 1, 2, \dots, \infty$

SUPERSCRIPTS

- 1 = indicates the coefficient of a previously existing term
- 2 = indicates the coefficient of a newly introduced term
- 12 = indicates the coefficient of the product of a previously existing term and a newly introduced term
- (n) = Fourier index
- $'$ = (prime), partial differentiation in the meridional direction
- \cdot = (dot), partial differentiation in the circumferential direction

ACKNOWLEDGEMENT

The author wishes to extend his finest appreciation to Dr. Robert E. Ball for his continuous and highly professional guidance and trust with "his program." The insight, foresight and encouragement provided by Dr. Ball were major factors leading to the timely completion of this effort and in increasing its worth. Appreciation is also extended to Drs. Johann Arbocz and Charles Babcock at the California Institute of Technology for the use of their experimental data.

The author also wishes to thank Drs. Richard F. Hartung, Bö O. Almroth, Frank A. Brogan and David Bushnell, and the Lockheed Missiles and Space Company's Palo Alto Research Laboratory , Structural Analysis Division for the personal education they imparted, for their aid in the preparation of the addendum to this thesis, and for the use of their structural analysis programs.

I. INTRODUCTION

Shells are used as structural elements in nearly every space vehicle, aircraft or other structure where light weight is of importance. Consequently, a large number of digital computer programs for the structural analysis of this shells have been developed. Many of these programs were briefly described by Bushnell in Ref. 1. An assessment concerning the analysis capability based on computational systems that have application to a broad class of shell structures was presented by Hartung in Ref. 2. In his assessment, Hartung discusses the limitations, scope, accuracy and efficiency of virtually every major shell structural analysis program currently available and under development. Of particular interest here, is the computer program SATANS -- Static And Transient Analysis, Nonlinear, Shells -- developed by Ball for the geometrically nonlinear analysis of arbitrarily loaded shells of revolution [Refs. 3 and 4].

First, in section II, a brief description of SATANS is presented including the basic theory, the program capabilities and limitations, the solution method and procedure and some general comments.

Second, in section III, SATANS-I is constructed. Modifications to SATANS making the program more user oriented are discussed. Included in these modifications are a reorganization of most of SATANS into a 'storable package,' incorporation of an equipment-independent plotting package

which uses a high-speed line printer, modification of iteration control, discussion of extrapolation technique and alteration of boundary load application procedures.

Third, and most important, the subject of imperfections is treated in section IV. Thin isotropic shells of revolution have consistently displayed experimental buckling loads much lower than the theoretically predicted buckling loads. This disagreement has generally been attributed to (1), the mathematical inability to correctly model the boundary conditions as they exist on the test specimen, and (2), a high degree of imperfection sensitivity, i.e., naturally-occurring imperfections in the initial shape of the shell cause a premature triggering of critical buckling modes and early shell collapse. An attempt by Dantone [Ref. 57] to modify SATANS and to include initial imperfections in the analysis of cylindrical shells was severely hampered by a restriction on the use of SATANS, namely, all applied loading and initial conditions -- and hence, initial imperfections -- must be symmetric about some meridional plane. Discussion of the removal of this restriction and introduction of initial imperfections into the program are covered in this section.

Fourth, and also in section IV, the implementation of the initial imperfection capability is presented. The resulting program is called SATANS-II. Steps for actual construction of SATANS-II from SATANS-I are presented as are instructions for its use.

Fifth, in section V, actual imperfect shell analyses are performed on two specific axially loaded circular cylinders. The actual topography of the cylinders has been measured by Arbocz and Babcock [Ref. 6] and this data provides the opportunity to include actual naturally-occurring initial imperfections into the analysis by SATANS-II. Detailed numerical studies are performed to determine the classical and consistent bifurcation buckling loads of the perfect cylinders and the results are compared with previously published results. Stability solutions for the imperfect cylinders under six boundary conditions are obtained and compared with experimental values. An analysis procedure for imperfect shells is proposed based upon these results.

Lastly, as an addendum to this report, the clamped shallow spherical cap is investigated under various boundary conditions at the pole. While on an industrial tour with the Structural Analysis Division of the Palo Alto Research Laboratory of the Lockheed Missiles and Space Company, Sunnyvale, California, Ryan noted that few shell structural analysis programs have the capability of analyzing a shell of revolution which has an initial or a final pole -- continuous structure through the axis of revolution. This investigation attempted to determine the effects on the solution caused by substitution of a small hole or a small rigid plug for the pole.

II. DESCRIPTION OF SATANS

The computer program known as SATANS is capable of performing static and transient analyses of thin, linearly-elastic shells of revolution subjected to arbitrary loads and temperature distributions. It is based on Sanders' nonlinear field equations [Ref. 7] for the conditions of small strains and moderately small rotations. The program can be used under the following conditions:

1. The geometric and material properties of the shell must be axisymmetric, but may vary along the shell meridian.
2. The shell material must be isotropic, but the modulus of elasticity may vary through the thickness.
3. Poisson's ratio must be constant.
4. The boundaries of the shell may be closed, fixed, elastically restrained or free, and loaded or unloaded.
5. The applied forces, pressures, temperatures and initial conditions must be symmetric about a datum meridional plane.

A. GEOMETRY AND COORDINATE SYSTEM

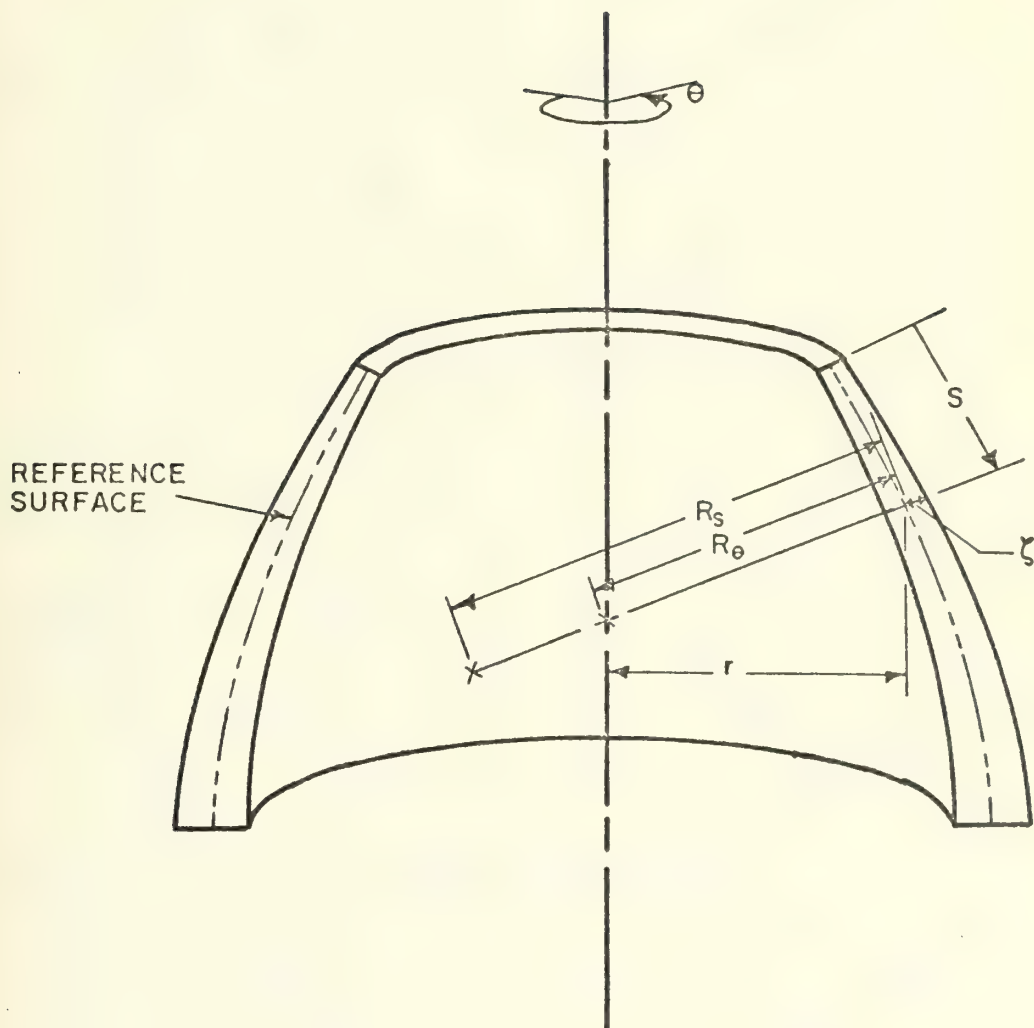
The coordinate system used to locate points within the shell is the s, θ, ζ orthogonal system, where s is the meridional distance along a reference surface, θ is the circumferential angle measured about the axis of revolution and ζ is the perpendicular distance from the reference surface. The shell geometry and coordinate system are shown

in Figure 1. Positive directions for displacements, rotations, forces, moments and loads are shown in Figure 2.

B. METHOD OF SOLUTION

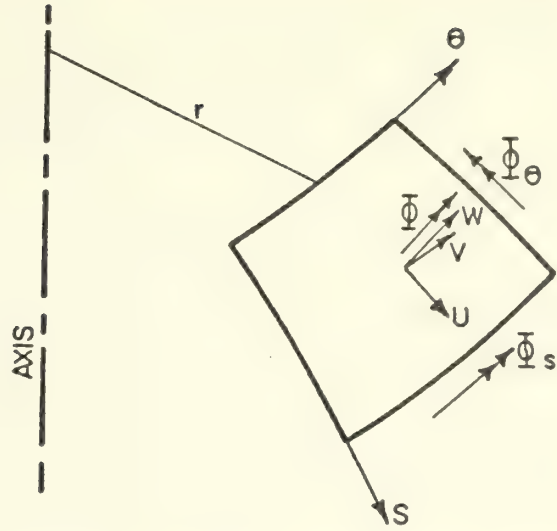
The governing partial differential equations are reduced to an infinite number of sets of four second order differential equations in the meridional and time coordinates by expanding all dependent variables in a sine or cosine series in terms of $n\theta$, where n is the Fourier index (harmonic) and θ is the circumferential coordinate. The sets are uncoupled by utilizing appropriate trigonometric identities and by treating the nonlinear coupling terms as 'pseudo loads.' The meridional derivatives are replaced by conventional central finite difference approximations, and the displacement accelerations are approximated by the implicit Houbolt backward differencing scheme [Ref. 8]. This leads to sets of algebraic equations in terms of the dependent variables and the Fourier index or harmonic. At each load or time step, an estimate of the solution is obtained by extrapolation from the solutions at previous load or time steps. The sets of algebraic equations are repeatedly solved using Potters' form of Gaussian elimination [Ref. 9]. The pseudo loads are recomputed after each solution and the procedure continues until the solution converges.

In the static analysis, a load-displacement history is obtained by an automatic incrementing of applied loadings in equal steps until the number of iterations required to achieve convergence in any harmonic exceeds a prescribed

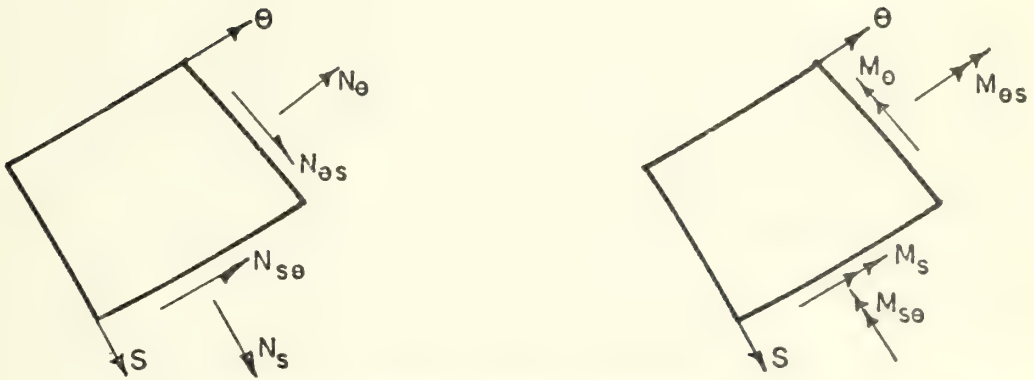


SHELL GEOMETRY AND COORDINATE SYSTEM

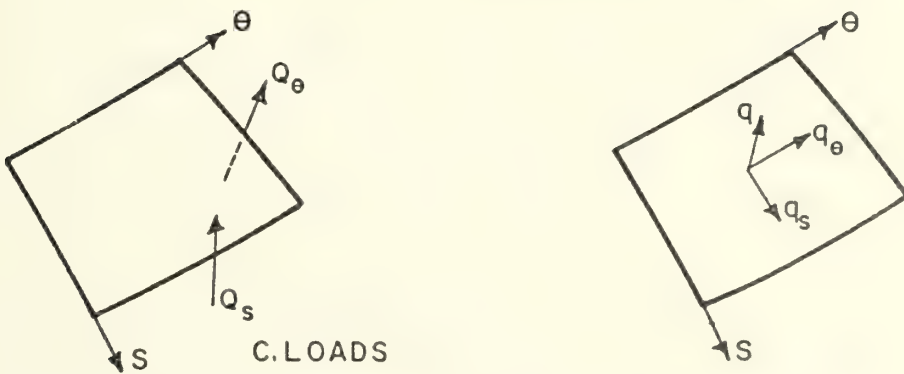
FIGURE 1



A. DISPLACEMENTS AND ROTATIONS



B. FORCES AND MOMENTS



C. LOADS

POSITIVE DIRECTIONS FOR VARIABLES

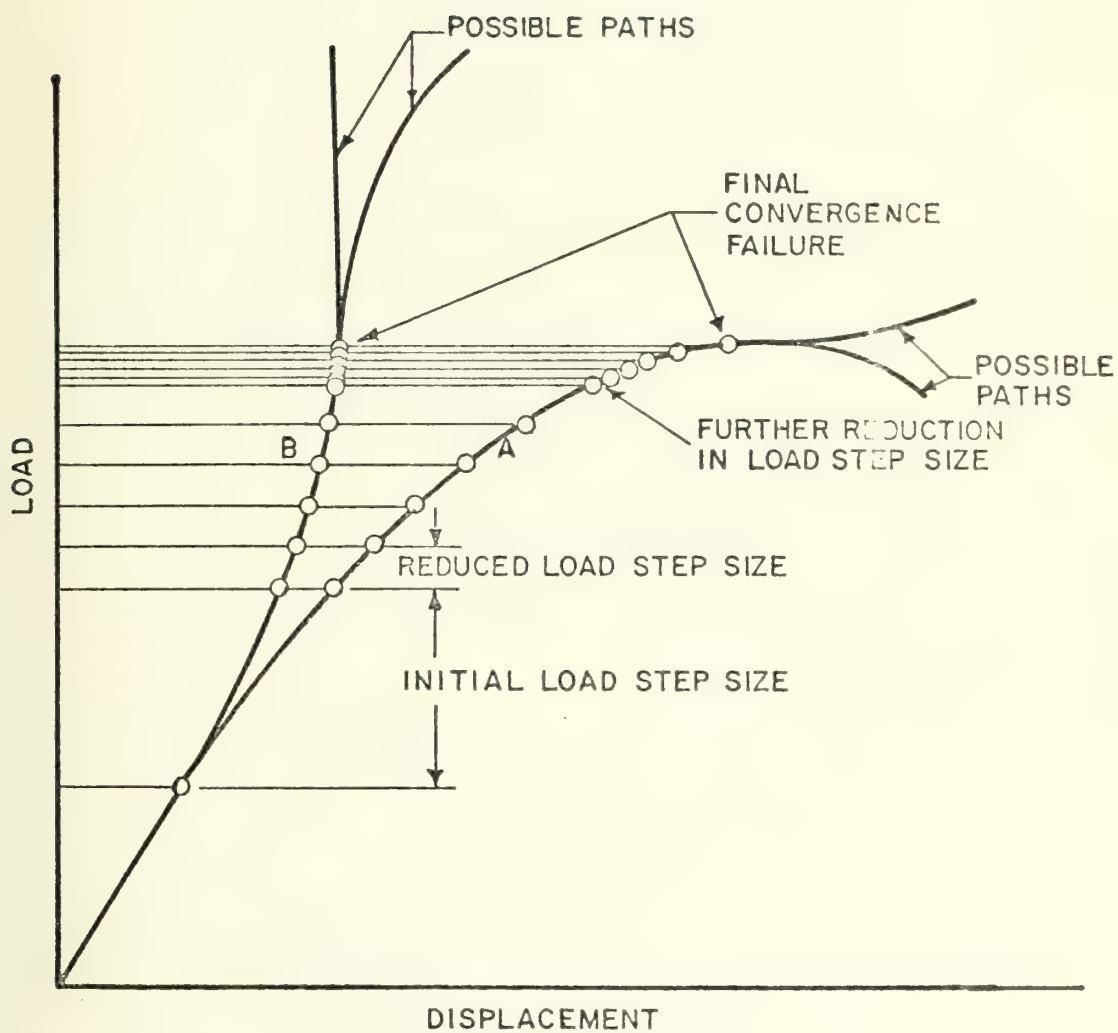
FIGURE 2

limit. The load increment is then reduced by a factor of five and the procedure continues until a prescribed number of load steps, or a prescribed number of load step reductions have been made.

Since the method of solution is based on the pseudo load approach, the shell reacts equally, in a linear fashion, to any change in either the applied load or the pseudo load. Thus, failure of the solution to converge in any harmonic can be attributed to two types of nonlinear behavior, both of which are illustrated in Figure 3. Referring to the figure, curve A experiences an inflection point or a maximum. Either condition causes a zero or nearly zero slope and a convergence failure. This curve is typical of softening nonlinearity. Curve B, on the other hand, is typical of a stiffening nonlinearity and a convergence failure will occur when the slope becomes large. It is necessary, then, to examine the load-displacement behavior of the shell to determine the cause of convergence failure. A detailed discussion of the theory and implementation of SATANS can be found on pages 8 through 29 and in Appendices B, C and D of Ref. 3, and on pages 8 through 28, pages 56 through 61 and page 100 of Ref. 4.

C. USER PREPARED SUBROUTINES

Five small subroutines describing the shell geometry, the stiffnesses, the loading (physical and/or thermal), and the initial conditions (in the dynamic case) must be prepared by the user. Complete descriptions for the



NONLINEAR BEHAVIOR AFFECTING CONVERGENCE

FIGURE 3

preparation of these subroutines can be found on pages 34 through 38 of Ref. 4.

III. CONSTRUCTION OF SATANS-I

Stein [Ref. 10] observes that a basic impediment to the widespread use of existing computer programs is the difficult and laborious process of assimilation of a program's basic theory, its implementation and its use by a prospective researcher. He also warns the 'unwary analyst' of the dangers inherent in computer output, which is a 'blizzard of numbers.' As a consequence, SATANS is continually being modified in an attempt to make it more usable. The most recent modifications made for this purpose are described in this section. This new version of SATANS is called SATANS-I.

A. PACKAGING

SATANS has been reorganized to allow 99% of the code to be prepackaged for repeated use by researchers. Compiled versions of all subroutines and main controlling routines can now be placed on a disk, in a data cell or on a magnetic tape, allowing the user to 'submit' only the user written subroutines and input data. A complete listing of this storable package is given in Appendix E.

B. PLOTTING ROUTINE

A great advantage is gained if the computer output can be presented in graphical format. Such plots, however, are often generated using mechanical or photo-electronic devices which are not universally usable. Consequently, SATANS has

been enhanced with an equipment-independent plot package utilizing self-contained subroutines and the high-speed line printer.

The main program listed in Ref. 4 could not be modified to incorporate the logic and control for a graphical ability as it then exceeded the capability of the FORTRAN-H compiler. Therefore, the main program of Ref. 4 has been replaced by a new main program and two subroutines described as follows:

SATANS - This is the new 'main.' It reads all data cards, prepares the title page of output and passes control to either of two basic controlling subroutines.

STATIC - This is the controlling subroutine for a static analysis. It conducts initialization, establishes geometry, stiffness, loading and boundary conditions, controls plotting of input information and controls the solution procedure.

DYNAMC - This is the controlling subroutine for a dynamic analysis and it operates in a fashion similar to subroutine STATIC.

The graphical ability is obtained by the introduction of two controlling subroutines, PLOT1 and PLOT2, four plotting subroutines, PLOTIT, SCALIT, ROUND and DRAWIT,¹ and modification of subroutine OUTPUT for the selection of output data for plotting. A single data card with request information

¹The four plotting subroutines are slightly modified versions of those contained in Ref. 11 under the listing "JS UTPLLOT."

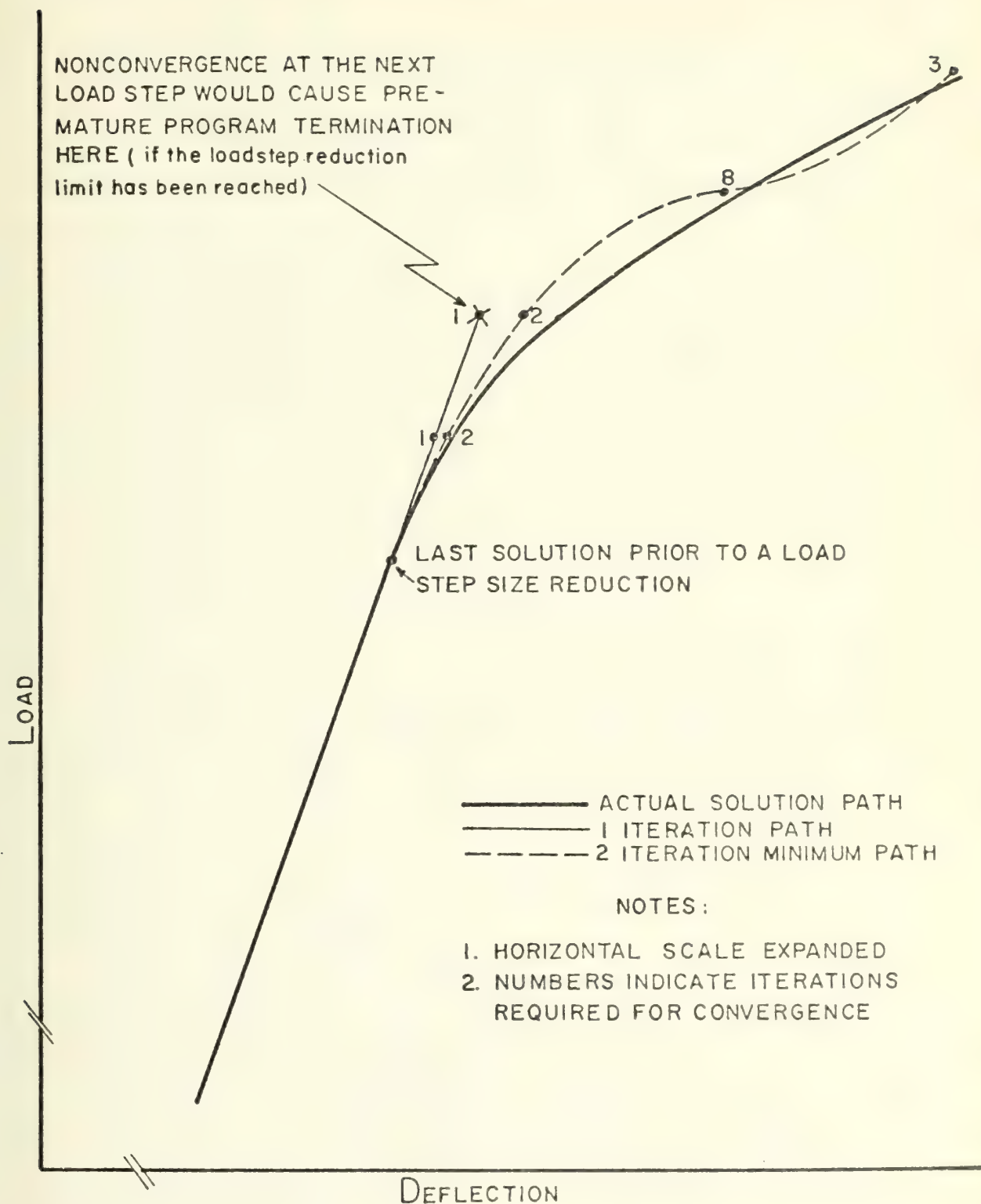
is required to generate the plots. The plotting routines will handle a maximum of 100 stations. This routine is included in the listing given in Appendix E.

C. MISCELLANEOUS ALTERATIONS

The iteration lower limit has been raised to two iterations. This change was introduced because it was noted that immediately after a load step size reduction, several consecutive solutions were obtained in one iteration, a completely linear process. This was then followed by non-convergence and, in some instances, premature program termination. Setting a minimum of two iterations automatically introduces nonlinearity into each load step and appears to solve this problem, as illustrated in Figure 4.

A quadratic extrapolation technique was investigated using a static analysis of the highly nonlinear shallow spherical cap. No gain in solution accuracy was seen and a price was paid in program execution time due to approximately 10% greater total iterations required, not to mention the increased storage space requirement. This phenomenon is exactly as observed by Stricklin et al [Ref. 127], hence the linear extrapolation technique has been retained.

Since boundary loads are introduced through the boundary conditions and since the program considers the boundary conditions to be the same in every harmonic, changes indicated in Appendix B of Ref. 5 are permanently incorporated into the program. This alteration causes the applied



PREMATURE PROGRAM TERMINATION

FIGURE 4

loading at a boundary to be taken in the zeroeth harmonic only, i.e., when $n=0$.

All 'COMMON' statements have been ordered and separated and some slightly modified. This alteration allows easier reading and correlation of FORTRAN variables with specific subroutines.

The output ability has been expanded to allow printed data (and plots) at up to 36 vice six meridians.

In the case of a shell with an initial (or a final) pole, determination of the shear force at the pole in the $n=1$ harmonic has been deemed an inconsequential and lengthy computation and has been removed. Appropriate changes have been made in subroutines INLPOL, FNLPOL and OUTPUT.

D. USE OF SATANS-I

Instructions for preparation of input data cards for SATANS-I are contained in Appendix C and the initiating main program and user written subroutines appear in Appendix D. Appendices D and E contain a complete listing of the computer program, SATANS-I.

SATANS-I requires 206 k-bytes of main core when run on an IBM-360/67 computer under OS/MVT and compiled by the FORTRAN-H compiler at optimization level 2. Execution times are the same as for SATANS unless plotting is requested, and then is increased approximately 0.01 seconds for each 100 point plot generated.

IV. CONSTRUCTION OF SATANS-II

As discussed previously, a serious impediment to the consideration of initial imperfections by Dantone [Ref. 57] was the requirement that the imperfections be symmetric about a datum meridian. This restriction results from the fact that all dependent variables were expanded in a single sine or cosine series. The analysis presented in this section removes this restriction by introducing the use of full trigonometric series expansions for the dependent variables, and then modifies the resulting new relations where required for the introduction of initial imperfections. The computer program that incorporates these changes is a modification to SATANS-I, called SATANS-II.

A. INTRODUCTION OF FULL TRIGONOMETRIC SERIES EXPANSIONS

The original assumption by Ball [Ref. 37] was that a normal pressure load could be expressed in a Fourier cosine series as

$$q = (\sigma_0 h_0 / E_0) \sum_{n=0}^{\infty} p^{(n)} \cos n\theta \quad (1)$$

where σ_0 = the reference stress level,

h_0 = the reference thickness,

E_0 = the reference Young's modulus,

n = the Fourier index, or harmonic,

and p = the nondimensional Fourier coefficient of the normal pressure load. The normal displacement W

corresponding to this pressure was taken as

$$W = (a\sigma_0/E_0) \sum_0^{\infty} w^{(n)} \cos n\theta \quad (2)$$

where a is the shell characteristic length and w is the nondimensional series coefficient of the normal displacement. For a completely arbitrary case, representing q and W with the complete Fourier series gives

$$q = (\sigma_0 h_0/E_0) \left(\sum_0^{\infty} p^1(n) \cos n\theta + \sum_1^{\infty} p^2(n) \sin n\theta \right) \quad (3)$$

and

$$W = (a\sigma_0/E_0) \left(\sum_0^{\infty} w^1(n) \cos n\theta + \sum_1^{\infty} w^2(n) \sin n\theta \right) \quad (4)$$

where superscript ¹ indicates a previously existing coefficient and superscript ² indicates a new coefficient. Similar series expansions can be written for the other pressures, displacements, etc.

The nonlinear terms associated with the full series are illustrated by ϕ_s^2 ,

$$\begin{aligned} \phi_s^2 = (\sigma_0/E_0)^2 & \left\{ \left(\sum_0^{\infty} \phi_s^1(n) \cos n\theta \right)^2 + \left(\sum_1^{\infty} \phi_s^2(n) \sin n\theta \right)^2 \right. \\ & \left. + 2 \left(\sum_0^{\infty} \phi_s^1(n) \cos n\theta \right) \left(\sum_1^{\infty} \phi_s^2(n) \sin n\theta \right) \right\} \quad (5) \end{aligned}$$

where the ϕ_s are the nondimensional series coefficients of

the meridional rotation, given as

$$\begin{aligned}\phi_S^1(n) &= (-w^1(n))' + \omega_S u^1(n) \\ \phi_S^2(n) &= (-w^2(n))' + \omega_S u^2(n)\end{aligned}\tag{6}$$

Originally, in Refs. 3 and 4, the nonlinear term ϕ_S^2 was represented as

$$\phi_S^2 = (\sigma_0/E_0)^2 \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right)^2 \equiv \sigma_0/E_0 \sum_0^\infty \beta_S^1(n) \cos n\theta \tag{7}$$

since

$$\cos(n\theta)\cos(m\theta) = \frac{1}{2}\cos(n-m)\theta + \frac{1}{2}\cos(n+m)\theta$$

The $\beta_S^1(n)$ term contains the products of all the ϕ_S 's whose harmonic sum or difference equals n . There are now two additional terms in equation (5). These can be represented as

$$(\sigma_0/E_0)^2 \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right)^2 \equiv \sigma_0/E_0 \sum_1^\infty \beta_S^2(n) \cos n\theta \tag{8}$$

and

$$\begin{aligned}2(\sigma_0/E_0)^2 \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right) \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right) \\ \equiv \sigma_0/E_0 \sum_1^\infty \beta_S^{12}(n) \sin n\theta\end{aligned}\tag{9}$$

where superscript 12 indicates the product of a previously existing coefficient and a new coefficient.

Thus,

$$\begin{aligned}\Phi_S^2 = \sigma_0/E_0 \cdot \left(\sum_0^{\infty} \beta_S^1(n) \cos n\theta + \sum_0^{\infty} \beta_S^2(n) \cos n\theta \right. \\ \left. + \sum_1^{\infty} \beta_S^{1,2}(n) \sin n\theta \right)\end{aligned}\quad (10)$$

Similarly, the nonlinear term Φ_θ^2 (and also Φ^2 since it is of the same form) is given by

$$\begin{aligned}\Phi_\theta^2 = (\sigma_0/E_0)^2 \left\{ \left(\sum_1^{\infty} \phi_\theta^1(n) \sin n\theta \right)^2 + \left(\sum_0^{\infty} \phi_\theta^2(n) \cos n\theta \right)^2 \right. \\ \left. + 2 \left(\sum_1^{\infty} \phi_\theta^1(n) \sin n\theta \right) \left(\sum_0^{\infty} \phi_\theta^2(n) \cos n\theta \right) \right\}\end{aligned}\quad (11)$$

which can be represented as

$$\begin{aligned}\Phi_\theta^2 = \sigma_0/E_0 \cdot \left(\sum_0^{\infty} \beta_\theta^1(n) \cos n\theta + \sum_0^{\infty} \beta_\theta^2(n) \cos n\theta \right. \\ \left. + \sum_1^{\infty} \beta_\theta^{1,2}(n) \sin n\theta \right)\end{aligned}\quad (12)$$

The nonlinear term $\Phi_S \Phi_\theta$ is given by

$$\begin{aligned}\Phi_S \Phi_\theta = (\sigma_0/E_0)^2 \left\{ \left(\sum_0^{\infty} \phi_S^1(n) \cos n\theta \right) \left(\sum_1^{\infty} \phi_\theta^1(n) \sin n\theta \right) \right. \\ + \left(\sum_0^{\infty} \phi_S^1(n) \cos n\theta \right) \left(\sum_0^{\infty} \phi_\theta^2(n) \cos n\theta \right) \\ + \left(\sum_1^{\infty} \phi_S^2(n) \sin n\theta \right) \left(\sum_1^{\infty} \phi_\theta^1(n) \sin n\theta \right) \\ \left. + \left(\sum_1^{\infty} \phi_S^2(n) \sin n\theta \right) \left(\sum_0^{\infty} \phi_\theta^2(n) \cos n\theta \right) \right\}\end{aligned}\quad (13)$$

which can be represented as

$$\begin{aligned} \phi_s \phi_\theta = \sigma_o/E_o \left(\sum_1^{\infty} \beta_{s\theta}^{1(n)} \sin n\theta + \sum_0^{\infty} \beta_{s\theta}^{12(n)} \cos n\theta \right. \\ \left. + \sum_0^{\infty} \beta_{s\theta}^{2(n)} \sin n\theta \right) \end{aligned} \quad (14)$$

Following this scheme, similar expressions can be obtained for the remaining nonlinear terms $\phi_s N_s$, $\phi_\theta N_\theta$, $\phi_s N_{s\theta}$, $\phi_\theta N_{\theta s}$, ϕN_s and ϕN_θ .

All of the nonlinear terms fall into one of the three categories $\cos(k\theta)\cos(l\theta)$, $\sin(k\theta)\sin(l\theta)$ or $\cos(k\theta)\sin(l\theta)$. Since these three forms were already accounted for in SATANS, there was no need to develop any new routines for the full series expansions. It was only necessary to introduce each new nonlinear term into the appropriate existing summation in subroutines PHIBET and TEAETA.

B. INTRODUCTION OF INITIAL IMPERFECTIONS

The initial imperfections only influence the nonlinear terms in the equations. The strain-displacement relationships used in this analysis are

$$\begin{aligned} \epsilon_s &= U' + W/R_s + \frac{1}{2}(\phi_s^2 + \phi^2) \\ \epsilon_\theta &= V'/r + r'U/r + W/R_\theta + \frac{1}{2}(\phi_\theta^2 + \phi^2) \\ \epsilon_{s\theta} &= \frac{1}{2}(V' + U'/r - r'V/r + \phi_s \phi_\theta) \end{aligned} \quad (15)$$

where

$$\begin{aligned}
 \phi_s &= -W' + U/R_s \\
 \phi_\theta &= -W^*/r + V/R_\theta \\
 \phi &= \frac{1}{2}(V' + r'V/r - U^*/r)
 \end{aligned} \tag{16}$$

and where ϵ_s , ϵ_θ , $\epsilon_{s\theta}$ represent the reference surface strains, ϕ_s , ϕ_θ , ϕ represent the reference surface rotations, and superscript primes and dots represent partial differentiation with respect to s and θ , respectively.

Consider a shell whose geometry differs slightly from a shell of revolution. For this imperfect shell, a deviation from a specific shape occurs in the form of a transverse displacement, \bar{W} . This \bar{W} produces no strain. Defining W^* as a total displacement,

$$W^* \equiv W + \bar{W} \tag{17}$$

and replacing W in equations (15) with W^* of equation (17) gives

$$\begin{aligned}
 \epsilon_s + \bar{\epsilon}_s &= U' + W^*/R_s + \frac{1}{2}(\phi_s^{*2} + \phi^2) \\
 \epsilon_\theta + \bar{\epsilon}_\theta &= V^*/r + r'U/r + W^*/R_\theta + \frac{1}{2}(\phi_\theta^{*2} + \phi^2) \\
 \epsilon_{s\theta} + \bar{\epsilon}_{s\theta} &= \frac{1}{2}(V' + U^*/r - r'V/r + \phi_s^*\phi_\theta^*)
 \end{aligned} \tag{18}$$

where

$$\begin{aligned}
 \phi_s^* &= -W^{*'} + U/R_s \\
 \phi_\theta^* &= -W^{*\cdot}/r + V/R_\theta
 \end{aligned} \tag{19}$$

and $\bar{\epsilon}_s$, $\bar{\epsilon}_\theta$ and $\bar{\epsilon}_{s\theta}$ are the pseudo strains due to \bar{W} only.

When $U = V = W = 0$, then $W^* = \bar{W}$, and $\epsilon_s \equiv \epsilon_\theta \equiv \epsilon_{s\theta} \equiv 0$.

Substituting these relationships into equation (18), solving for $\bar{\epsilon}_s$, $\bar{\epsilon}_\theta$, $\bar{\epsilon}_{s\theta}$ and placing these terms in equations (16) lead to

$$\epsilon_s = U' + W^*/R_s + \frac{1}{2}(\phi_s^{*2} + \phi^2) - \bar{W}/R_s - \frac{1}{2}(\bar{W}')^2 \quad (20)$$

$$\epsilon_\theta = V'/r + r'U/r + W^*/R_\theta + \frac{1}{2}(\phi_\theta^{*2} + \phi^2) - \bar{W}/R_\theta - \frac{1}{2}(\bar{W}')^2/r$$

$$\epsilon_{s\theta} = \frac{1}{2}(V' + U'/r - r'V/r + \phi_s^*\phi_\theta^* - \bar{W}'\bar{W}^*/r)$$

Expansion of equations (20) making use of equations (16), (17) and (19) lead to

$$\epsilon_s = U' + W/R_s + \frac{1}{2}(\phi_s^2 + \phi^2) - \{\bar{W}'\phi_s\}$$

$$\epsilon_\theta = V'/r + r'U/r + W/R_\theta + \frac{1}{2}(\phi_\theta^2 + \phi^2) - \{\bar{W}'\phi_\theta/r\} \quad (21)$$

$$\epsilon_{s\theta} = \frac{1}{2}(V' + U'/r - r'V/r + \phi_s\phi_\theta - \{\bar{W}'\phi_\theta + \bar{W}'\phi_s/r\})$$

where the terms enclosed in braces are the newly generated terms due to the imperfections, and

$$\phi_s^* = \phi_s - \{\bar{W}'\} \quad (22)$$

$$\phi_\theta^* = \phi_\theta - \{\bar{W}'/r\}$$

The presence of imperfections has no effect on the changes of curvatures since the curvature-displacement relations are linear.

Expressing the imperfection partial derivatives $-\bar{W}'$ and $-\bar{W}'/r$ as imperfection rotations $\bar{\Phi}_S$ and $\bar{\Phi}_\theta$, the strain-displacement equations (21) can be expressed as

$$\begin{aligned}\epsilon_S &= U' + W/R_S + \frac{1}{2}(\phi_S^2 + \{2\phi_S \bar{\Phi}_S\} + \phi^2) \\ \epsilon_\theta &= V'/r + r'U/r + W/R_\theta + \frac{1}{2}(\phi_\theta^2 + \{2\phi_\theta \bar{\Phi}_\theta\} + \phi^2) \quad (23) \\ \epsilon_{S\theta} &= \frac{1}{2}(V' + U'/r - \phi_S \phi_\theta + \{\phi_\theta \bar{\Phi}_S + \phi_S \bar{\Phi}_\theta\})\end{aligned}$$

where the terms in braces are the newly generated terms. Imperfection rotations also affect the nonlinear term, $\phi_S N_S$ (and the others listed after equation (14)). For these terms, the reference surface rotations ϕ_S and ϕ_θ must be replaced by ϕ_S^* and ϕ_θ^* of equations (22).

To illustrate the procedure which is used in the following section to introduce the initial imperfection rotations, consider the single series expansion, equation (7), repeated here,

$$\phi_S^2 = (\sigma_0/E_0)^2 \left(\sum_0^\infty \phi_S^{1(n)} \cos n\theta \right)^2 \equiv \sigma_0/E_0 \sum_0^\infty \beta_S^{1(n)} \cos n\theta \quad (7)$$

Modifying this equation to include the effect of initial imperfections indicated by equations (23) results in

$$\begin{aligned}\phi_S^2 + \{2\phi_S \bar{\Phi}_S\} &= (\sigma_0/E_0)^2 \left[\left(\sum_0^\infty \phi_S^{1(n)} \cos n\theta \right)^2 \right. \\ &\quad \left. + 2 \left(\sum_0^\infty \phi_S^{1(n)} \cos n\theta \right) \left(\sum_0^\infty \bar{\Phi}_S^{1(n)} \cos n\theta \right) \right] \quad (24) \\ &\equiv \sigma_0/E_0 \sum_0^\infty \beta_S^{*1(n)} \cos n\theta\end{aligned}$$

A redefinition of β_s to include the cross-product term is now required, along with redefinitions of the other β terms and the η terms in the other nonlinear term expressions. A redefinition of these terms alone is sufficient for the introduction of initial imperfections into a SATANS analysis where only the single series expansions are used. This was the thrust of the thesis by Dantone.

C. COMBINATION OF FULL TRIGONOMETRIC SERIES EXPANSIONS AND INITIAL IMPERFECTIONS

Consider that for every meridional rotation ϕ_s , where

$$\phi_s = (\sigma_0/E_0)^2 \left(\sum_0^{\infty} \phi_s^1(n) \cos n\theta + \sum_1^{\infty} \phi_s^2(n) \sin n\theta \right) \quad (25)$$

there exists an initial meridional imperfection rotation $\bar{\phi}_s$, defined as

$$\bar{\phi}_s = (\sigma_0/E_0)^2 \left(\sum_0^{\infty} \bar{\phi}_s^1(n) \cos n\theta + \sum_1^{\infty} \bar{\phi}_s^2(n) \sin n\theta \right) \quad (26)$$

The nonlinear term ϕ_s^2 , previously given by equation (5), now becomes $(\phi_s^2 + 2\phi_s\bar{\phi}_s)$, according to equation (23), where ϕ_s^2 is given by equation (5) and $\phi_s\bar{\phi}_s$ is the product of equations (25) and (26), as illustrated for the single series expansion in equation (24). All the nonlinear terms of equations (15) have been converted to those in equation (23) for the full series expansions and the resulting relations are given in Appendix A. The definitions of the β terms (given previously in Ref. 3) have been modified to include the

effects of both full series expansions and initial imperfection presence.

The remaining nonlinear terms (those listed after equation (14)) also require modification but are of simpler form since only the first power of the rotation is involved. A redefinition of the η terms to include the effects of full series expansions and appropriate initial imperfection rotations has also been made.

In this manner, the validity of the strain-displacement equations (equations (5) of Ref. 3 and equations (15) of this report) and the equilibrium equations (equations (8) of Ref. 3) is retained. Introduction of full series expansions and initial imperfections in the nonlinear terms was accomplished by a redefinition of the β and η terms, which are the nondimensional coefficients for the nonlinear terms in the strain-displacement and equilibrium equations, respectively.

D. SUBROUTINE MODIFICATIONS

Introduction of the full series into the program was implemented by use of negative index numbers for the new sine or cosine portions of the expansions (the terms with the superscript ²). With the requirement that the axisymmetric harmonic ($n=0$) be included, the program was expanded to handle a maximum of 25 modes, i.e., the zeroeth harmonic ($n=0$) plus twelve other harmonics (positive and negative). Each positive index and each negative index will be referred to as a mode, i.e., the harmonic $n=5$ has two modes, -5

and +5. A typical correspondence between harmonic, mode and Fourier index is given in Table I. Harmonic zero ($n=0$) and any combination of up to twelve harmonics may be included in the analysis. The positive values of NN in Table I are the modes analyzed in SATANS and SATANS-I. The negative values of NN are those modes added to incorporate the full series expansions in SATANS-II. Note that $n=0$ is the first mode, and that the negative value is always the first mode of a pair of modes. This is a requirement.

TABLE I

HARMONIC, MODE AND FOURIER INDEX CORRESPONDENCE

HARMONIC n	MODE	FOURIER INDEX	FORTRAN ASSIGNMENT STATEMENT IN SUBROUTINE PLOAD OR TLOAD
0	1	0	NN(1)=0
1	2	-1	NN(2)=-1
	3	+1	NN(3)=+1
2	4	-2	NN(4)=-2
	5	+2	NN(5)=+2
5	6	-5	NN(6)=-5
	7	+5	NN(7)=+5
.	.	.	.
.	.	.	.
.	.	.	.
18	24	-18	NN(24)=-18
	25	+18	NN(25)=+18

Seven subroutines of SATANS-I required changes to incorporate the full trigonometric series expansions and the initial imperfections.

Subroutine PHIBET calculates the ϕ 's and carries out the summation procedures for calculating the β nonlinear terms. Since extensive modification was required, the subroutine was altered such that those sections handling the full trigonometric expansions and imperfections are entered only if required, i.e., if a full series analysis is necessary.

Subroutine TEAETA calculates the inplane forces and the η nonlinear terms. It was also extensively modified, again, utilizing new sections only if required.

The subroutine MODES, a highly complex procedure for determining modal coupling required minimal alteration since each unique mode number is stored at every, or every other storage location, depending on use of SATANS-I or SATANS-II, respectively. Each unique harmonic is accessed, then, by an appropriate step size change in the controlling loops.

The subroutine OUTPUT required the addition of a section to include the new half of the trigonometric series in the generation of a summed solution at a specified meridian.

The subroutines POLE, INLPOL and FNLPOL compute the solution at an initial or a final pole. Due to the computation of pole conditions based on n , which now has positive and negative values, these routines were modified

accordingly. Effectively, the only change in the equations presented in Appendix D of Ref. 3 involve a replacement of n by $-n$ where appropriate.

A complete listing of these seven modified subroutines is given in Appendix F.

E. SATANS-II

To modify SATANS-I to handle completely arbitrary loading and initial imperfections, the following steps must be performed.

1. Replace subroutines PHIBET, TEAETA, MODES, OUTPUT, POLE, INLPOL and FNLPOL with those in Appendix F.
2. Replace the affected COMMON statements in Appendices D and E with those listed in Appendix F.
3. Add subroutine IMPERF, also listed in Appendix F. This subroutine contains the values of the imperfection rotation coefficients, $\bar{\phi}_s$ and $\bar{\phi}_\theta$. Instructions for actual preparation of these coefficients will be discussed in detail in section V. If imperfections are not to be considered, the $\bar{\phi}_s$ and $\bar{\phi}_\theta$ are never accessed.
4. In Appendix E, make the following additions after the card indicated:

<u>Subroutine</u>	<u>After Card Number</u>	<u>Insert</u>
DYNAMC	00002270	CALL IMPERF
STATIC	00002230	CALL IMPERF
PMATRX	00001100	IF(N(MN).LT.0)P(2,1,IJ)=-1.0
PMATRX	00002000	IF(N(MN).LT.0)ZFP1(2,2)=1.0
SATANS	00000380	READ(5,108)JUMP,MPERFS

SATANS 00001400 108 FORMAT(2I5)

5. In Appendix E, modify the following cards:

<u>Subroutine</u>	<u>Card Number</u>	<u>Should Read</u>
DYNAMC	00000800	DIMENSION VBAR(25),AVB(25)
PMATRX	00000900	NN=IABS(N(MN))
PMATRX	00001640	NN=IABS(N(MN))

6. In Appendix C, modify the data preparation instructions by adding a data card after card #4 according to the following format:

<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Interpretation</u>
4.1	1-5	I5	JUMP	1 = single series expansion 2 = double series expansion
4.1	6-10	I5	MPERFS	0 = no imperfections 1 = imperfections

An initial imperfection solution requires the use of full series in the analysis. Specification of the parameter JUMP indicates whether the solution will be as in SATANS-I or will utilize the full trigonometric series expansions of SATANS-II. Specification of the parameter MPERFS indicates whether the analysis will include initial imperfections. Note that if MPERFS equals 1, JUMP must equal 2, but if JUMP equals 2, MPERFS may be either 1 or 0.

7. Add imperfection data just prior to the last card of the data deck. This data is to be placed in arrays PHIXB and PHITB.

F. USE OF SATANS-II

After completing the changes in section IV.E, SATANS-II exists as a program capable of performing all the functions of SATANS-I with the following restrictions:

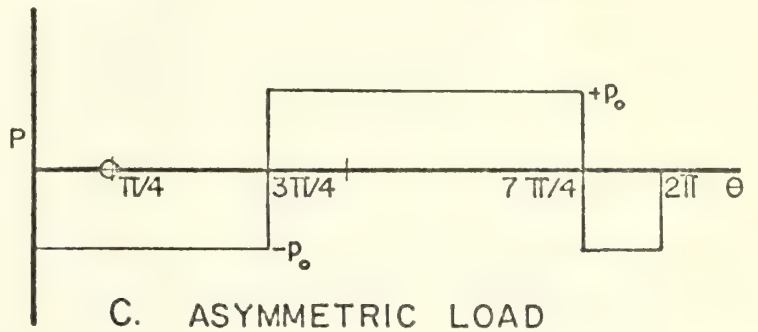
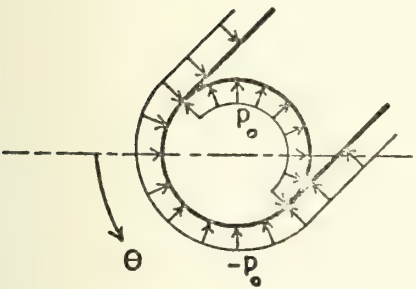
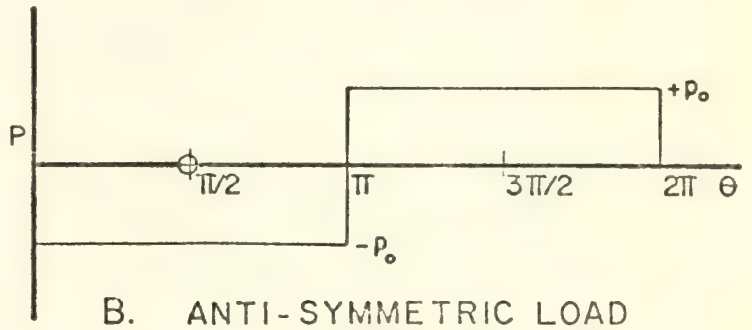
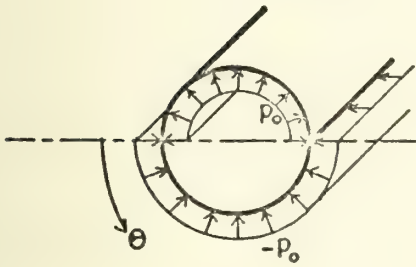
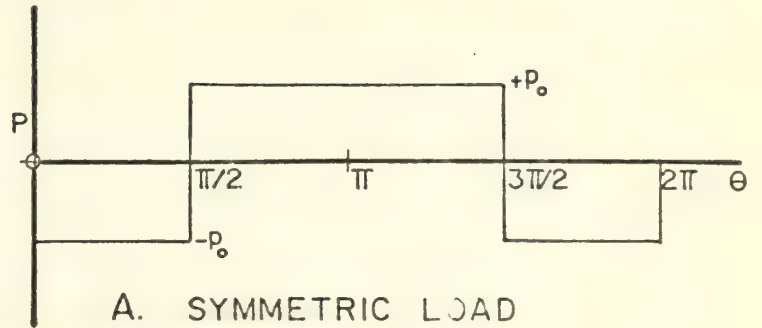
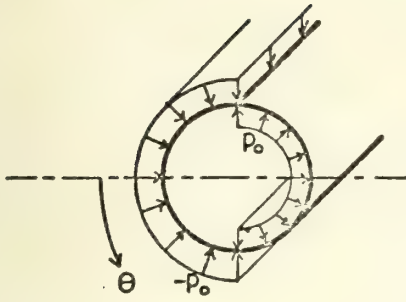
1. The number of meridional stations cannot exceed 100, ($KMAX < 101$).
2. The number of modes cannot exceed 25, ($MNMAX < 26$).
3. The product of meridional stations and modes cannot exceed 951, ($KMAX * MNMAX < 952$).
4. Plotting ability is not available for the imperfection input data.

Specification of harmonics to be considered (accomplished in subroutine PLOAD or TLOAD) must include the axisymmetric harmonic ($n=0$) and both negative and positive entries for any other harmonic, as indicated in Table I.

The instructions for the preparation of the input data are given in Appendix C. SATANS-II requires 426 k-bytes of main core on an IBM-360/67 computer when run under OS/MVT and when compiled by the FORTRAN-H compiler at optimization level 2. Execution times vary widely from four minutes for a 35 station and 3 mode analysis, up to 30 - 60 minutes for a 35 station and 25 mode analysis.

G. VALIDITY OF SATANS-II

SATANS-II was checked for its ability to obtain a correct solution using the full series expansions. This check was made using a clamped cylindrical shell subjected to pressure loadings as described in Figure 5. A solution was



\oplus = SOLUTION COMPARISON POINTS

SATANS-II CHECK LOADINGS
FIGURE 5

obtained for the loading shown in Figure 5A with SATANS-I and the summed solution was printed at meridian $\theta = 0^\circ$. The cylinder was then analyzed by SATANS-II for the following three orientations:

1. Loading defined as in Figure 5A - a symmetric load.
2. Loading defined as in Figure 5B - an anti-symmetric load. In this case, the summed solution was printed at meridian $\theta = 90^\circ$.
3. Loading defined as in Figure 5C - a general load. In this case, the summed solution was printed at meridian $\theta = 45^\circ$.

Load definitions for the four check runs appear in Table II. In each case, the solution obtained by SATANS-II

TABLE II

FOURIER COEFFICIENTS FOR CHECK LOADINGS

MODE	FOURIER COEFFICIENTS OF APPLIED LOADING		
	SYMMETRIC	ANTI-SYMMETRIC	NONSYMMETRIC
0	0.0	0.0	0.0
-1	0.0	-63.66	-45.015
+1	-63.66	0.0	-45.015
-3	0.0	+21.22	+15.005
+3	+21.22	0.0	-15.005
-5	0.0	-12.73	+9.003
+5	-12.73	0.0	+9.003
-7	0.0	+9.095	-6.431
+7	+9.095	0.0	+6.431

Note: the coefficients are based on $p_0 = 50$ psi.

agreed with the SATANS-I solution² through four significant digits.

²This reference to solution includes the load-deflection history, the ultimate nonconvergence level, and the displacement, force and moment distributions.

V. IMPERFECT SHELL ANALYSIS

A. PREPARATION OF THE IMPERFECTION DATA

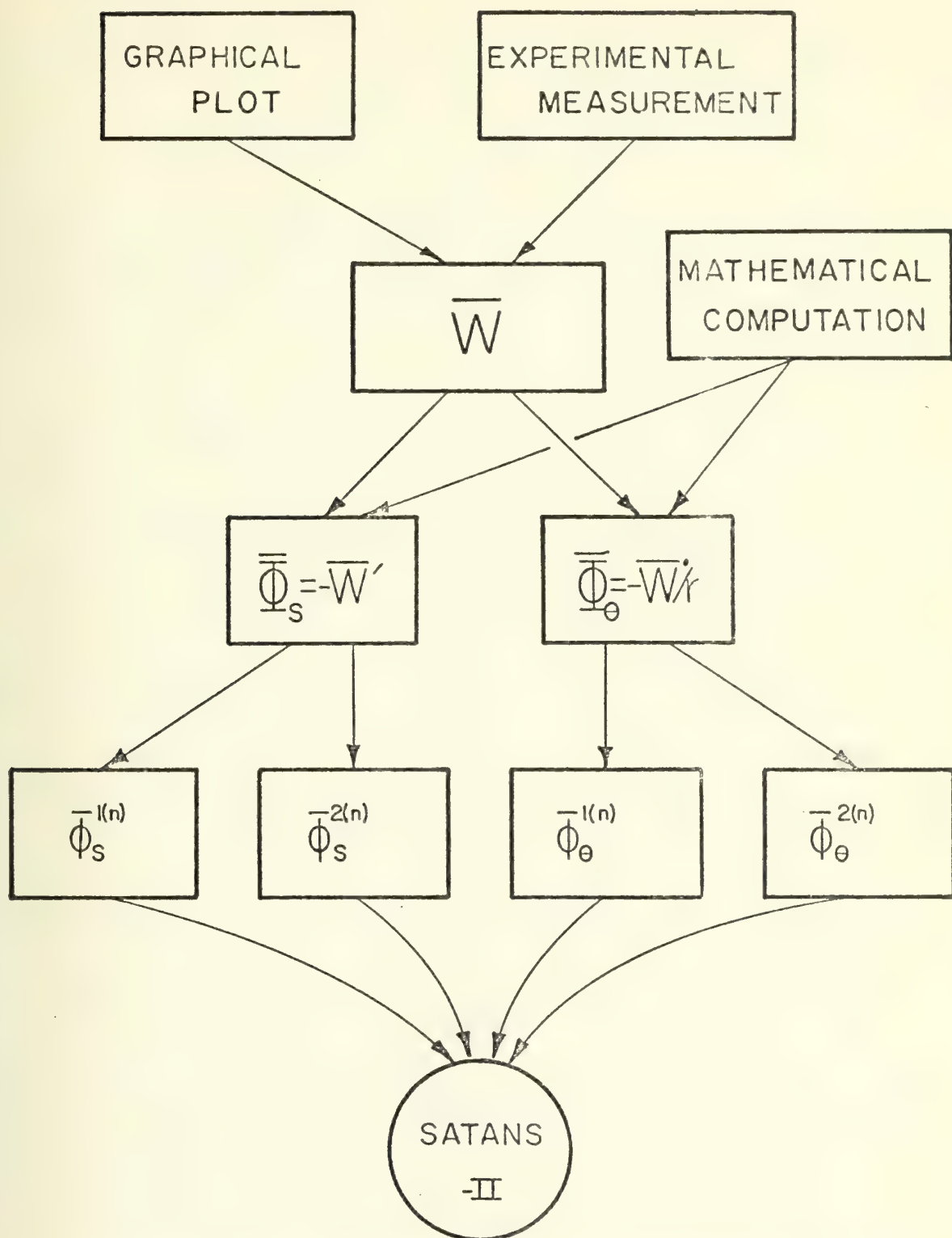
With the developments of section IV implemented, SATANS-II requires only the Fourier coefficients of the imperfection data to conduct a nonlinear analysis. Recall from equation (26) that these Fourier coefficients, $\bar{\phi}_s$ and $\bar{\phi}_\theta$ are determined by knowledge of the actual imperfection rotations, $\bar{\phi}_s$ and $\bar{\phi}_\theta$. Recall also from equation (22) that the imperfection rotations are given as

$$\begin{aligned}\bar{\phi}_s &= -\bar{W}' \\ \bar{\phi}_\theta &= -\bar{W}''/r\end{aligned}\tag{27}$$

This means that the imperfection rotation Fourier coefficients can be determined by knowledge of the initial normal deflection deviation, \bar{W} . This imperfection information can be obtained from a mathematical description, a graphical plot or from actual experimental measurement. A simplified data generation flow chart is presented in Figure 6.

1. Data Generation by Mathematical Description

In a note on imperfect shallow spherical cap buckling, Kao [Ref. 13] analyzes a cap subjected to a mathematically defined dimple imperfection existing over one-quarter of the shell. The dimple is given by



SIMPLIFIED IMPERFECTION DATA GENERATION
SCHEME

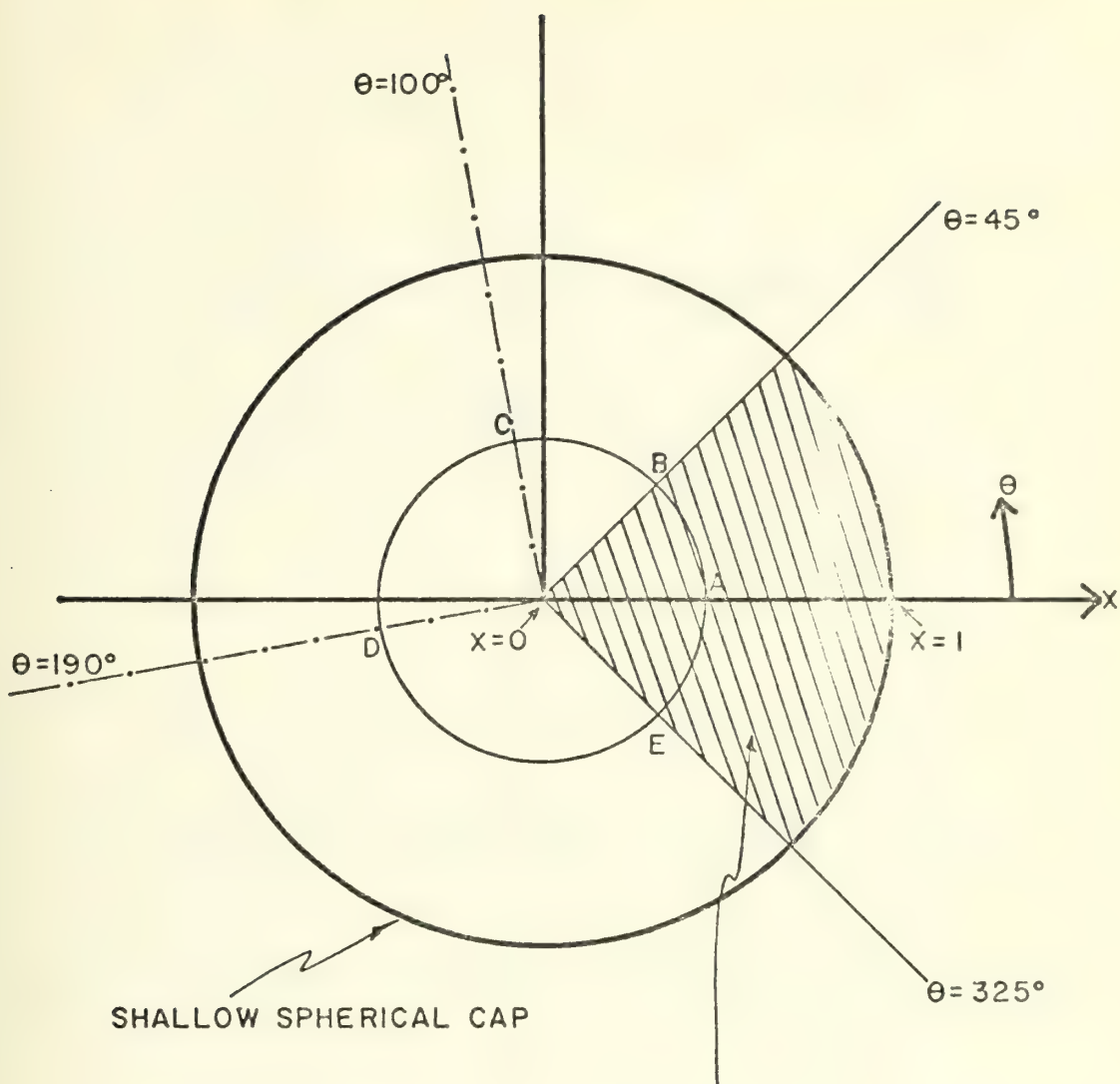
FIGURE 6

$$\bar{W} = 64(s/L)^3(1-s/L)^3 h \delta \cos 2\theta, \quad -45^\circ < \theta < 45^\circ$$

$$\bar{W} = 0, \quad 45^\circ < \theta < 325^\circ$$
(28)

where L is the total length of the shell meridian and where s/L ranges from 0 at the pole to 1 at the edge, and δ is an intensity factor, $0 \leq \delta \leq 1$. The geometry of this problem is shown in Figure 7. Kao used a two-dimensional finite difference mesh of eight meridional and 16 circumferential stations in his analysis. A numerical model of, for example, 35 meridional stations and 25 modes would yield a SATANS-II solution of considerable accuracy. The discrete initial normal deflection deviations \bar{W} at each mesh point need not be computed, as partial differentiations of equation (28) give expressions directly solvable for discrete values of $\bar{\Phi}_s$ and $\bar{\Phi}_\theta$ for each s and θ . Imperfection rotations at locations exterior to the affected area ($-45^\circ < \theta < 45^\circ$) are zero. To illustrate, the meridional imperfection rotations that exist along the line ABCDEA of Figure 7 would appear as curve A of Figure 8. The discrete values generating this curve can easily be expanded into a full Fourier series and the resulting coefficients $\bar{\Phi}_s^{(n)}$ inserted into SATANS-II for the analysis.

The imperfection of Figure 7 is symmetric about the meridian $\theta = 0^\circ$. As an example of a more complex imperfection, another dimple of the same form could be located in the region defined by $100^\circ < \theta < 190^\circ$. In this asymmetric case, the meridional imperfection rotations along line



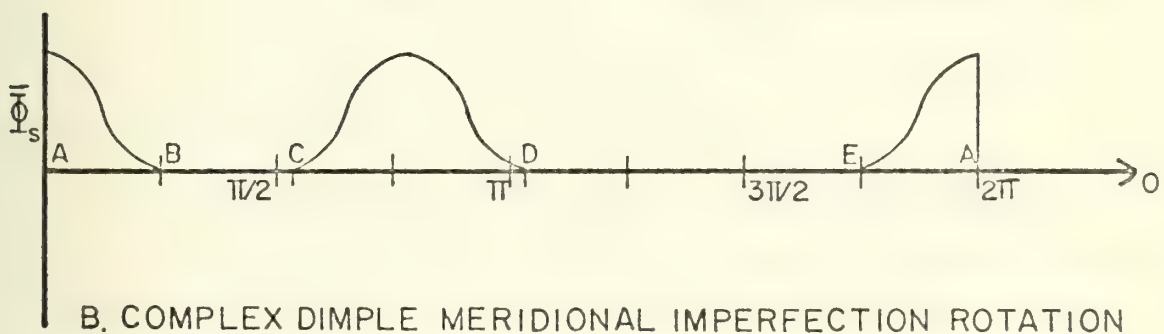
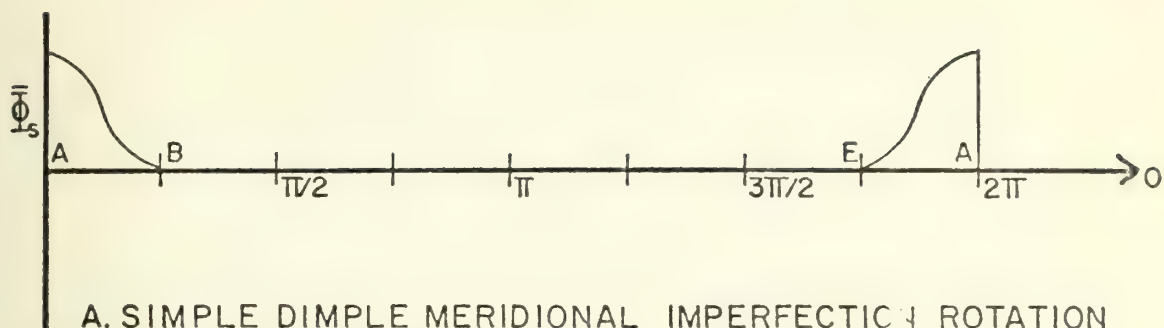
IMPERFECT QUARTER WITH
DIMPLE IMPERFECTION GIVEN BY

$$\bar{W}/h = 64x^3(1-x)^3 \int \cos 2\theta$$

WHERE $x = s/L$

MATHEMATICAL IMPERFECTION GENERATION

FIGURE 7



MERIDIONAL IMPERFECTION ROTATION REPRESENTATION

FIGURE 8

ABCDEA of Figure 7 would now appear as curve B of Figure 8. The discrete values generating this curve can also be easily expanded into the full trigonometric series and its coefficients inserted into SATANS-II.

2. Data Generation by Graphical Plots

Graphical representations of \bar{W} can be easily constructed and the patterns are near limitless in design or location. Theoretical or actual creases, furrows or dents of any shape need only be graphed, the discrete imperfection displacements determined, the discrete imperfection rotations $\bar{\Phi}_s$ and $\bar{\Phi}_\theta$ computed via a finite differencing technique, the Fourier expansions performed and the resulting coefficients inserted into SATANS-II.

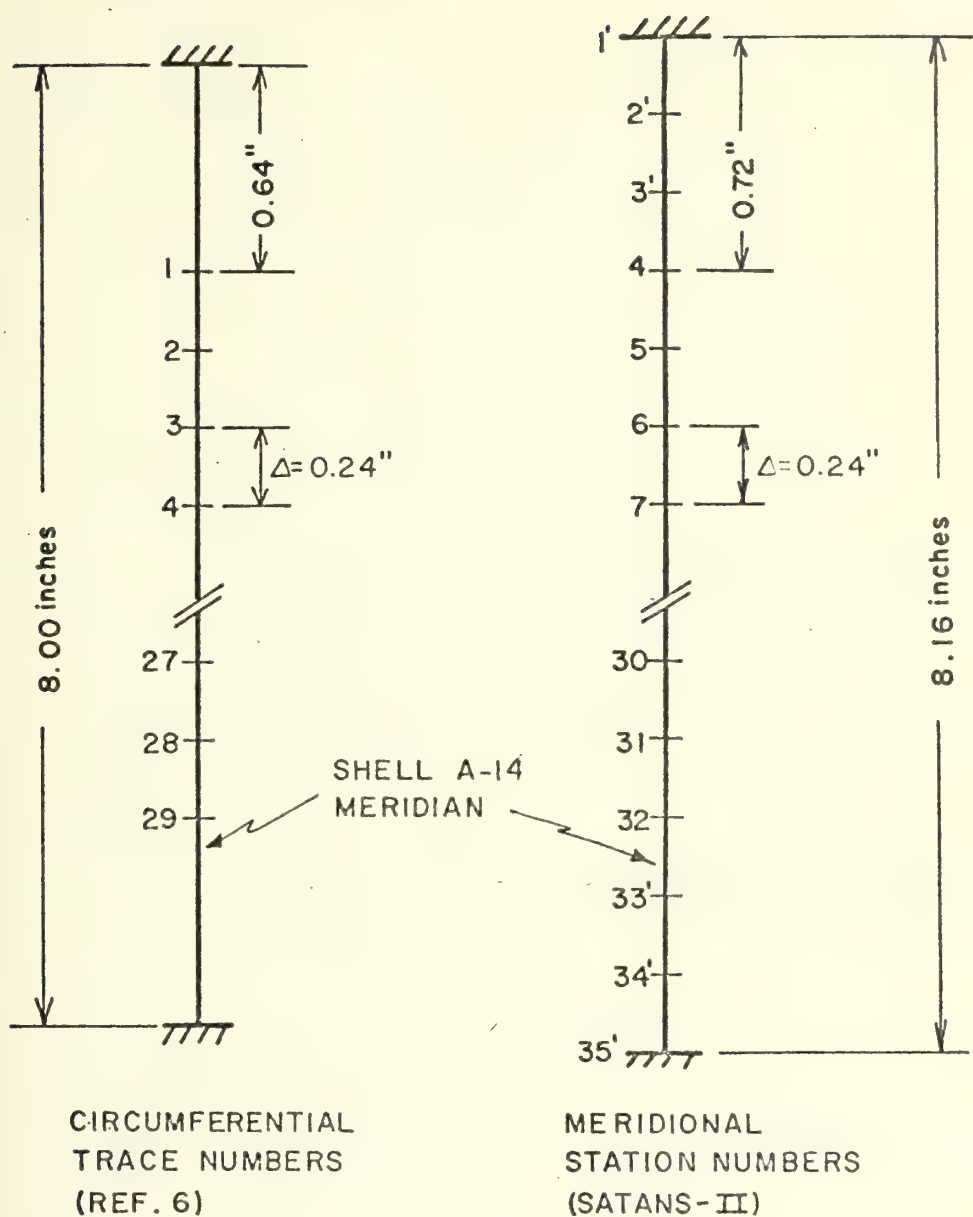
3. Data Generation by Experimental Measurement

Arbocz and Babcock [Ref. 6] have developed a means of mapping the topography of circular cylindrical shells, and can thus describe the deviations of an actual shell. The procedure, described in detail in Ref. 6, produces imperfection data in the form of nondimensional normal displacements from a perfectly circular cylinder. For the sample shells considered here, 49 data points were taken around each circumference and 29 equally-spaced circumferential traces were made. The first and last trace were 0.64 inches from the ends of the shell and the spacing between traces was 0.24 inches. Pertinent geometrical and material data concerning the shells of interest here is contained in Table III.

TABLE III
IMPERFECT SHELL OVERALL CHARACTERISTICS

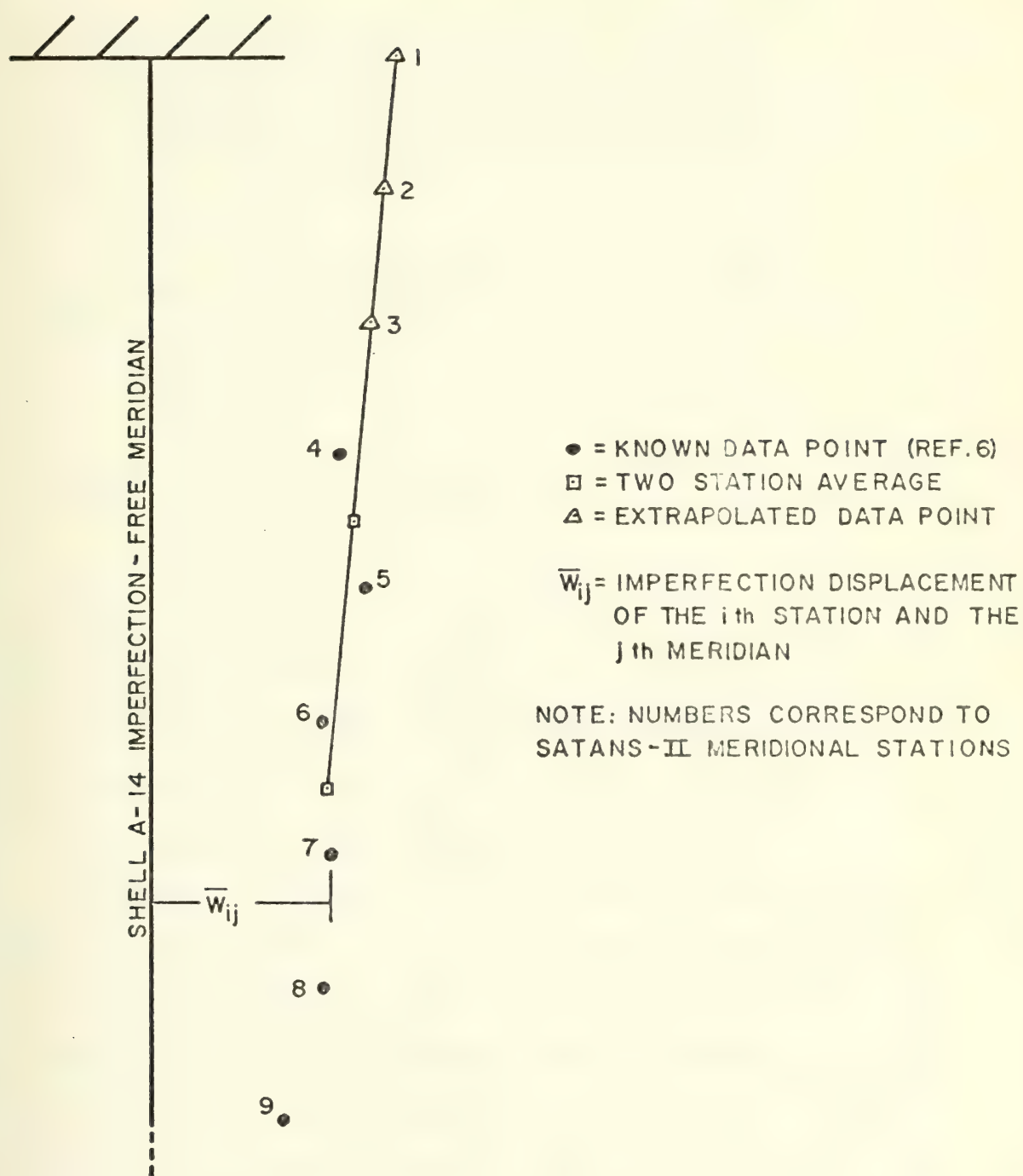
SHELL DESIGNATION	A-13	A-14
Radius	4.00"	4.00"
Length	8.00"	8.00"
Young's Modulus $\times 10^6$, E	15.10 psi	15.80 psi
Wall thickness, h	0.00444"	0.00437"
Number of axial traces	29	29
Axial increment, Δ	0.24"	0.24"
Number of data points	1421	1421

To prepare the imperfection data on shell A-14 for use by SATANS-II, the nondimensional deviations were first dimensionalized by the wall thickness. To retain the original axial increment of 0.24 inches between circumferential traces and to make the data compatible with a 35-station meridian, the shell length was increased by 0.08 inches at each end, or a total of 2% (See Figure 9). This required three additional traces at each end of the shell corresponding to stations 1, 2 and 3, and 33, 34 and 35. Data for the additional six circumferential traces were generated via the modified linear extrapolation procedure illustrated in Figure 10. At each of the 1,715 data points, the meridional and circumferential rotations were computed



CIRCUMFERENTIAL TRACE NUMBER AND
MERIDIONAL STATION NUMBER CORRESPONDENCE

FIGURE 9



DATA POINT EXTRAPOLATION TECHNIQUE FOR \bar{w}

FIGURE 10

using central finite differencing in the appropriate direction. After nondimensionalization according to the SATANS-II scheme, each circumferential vector of 48 meridional and 48 circumferential imperfection rotations was expanded via a periodogram into a Fourier series according to

$$(E_0/\sigma_0)\bar{\phi}_S = \bar{\phi}_S(0) + \sum_1^{24} \bar{\phi}_S^C(n) \cos n\theta + \sum_1^{24} \bar{\phi}_S^S(n) \sin n\theta \quad (29)$$

and

$$(E_0/\sigma_0)\bar{\phi}_\theta = \bar{\phi}_\theta(0) + \sum_1^{24} \bar{\phi}_\theta^C(n) \cos n\theta + \sum_1^{24} \bar{\phi}_\theta^S(n) \sin n\theta \quad (30)$$

At this point, the Fourier coefficients $\bar{\phi}_S$ and $\bar{\phi}_\theta$ are of proper form for direct insertion into SATANS-II. The computer program which performed the actual imperfection data processing for shell A-14 is given in Appendix G.

4. Selection of Important Harmonics

Since the number of meridional stations was 35, the program maximum of 25 modes can be used in the analysis.³ Note, however, that in equations (29) and (30), there are 49 modes describing the imperfections. Therefore, a selection of a maximum of 25 imperfection modes must be made. The harmonics to be considered in the analysis should include those critical to the shell and those significant in the imperfection. The description of the harmonic

³Recall the restrictions given in section IV.F.

selection process for shell A-14 is given in a later section.

B. BOUNDARY CONDITIONS

The testing configuration used by Arbocz and Babcock on the axially loaded cylinder is shown in Figure 11. The ends of the shell are attached to an end ring and a load cell with cerrelow, a low melting-point alloy. The ends of the shell are believed to be clamped so that

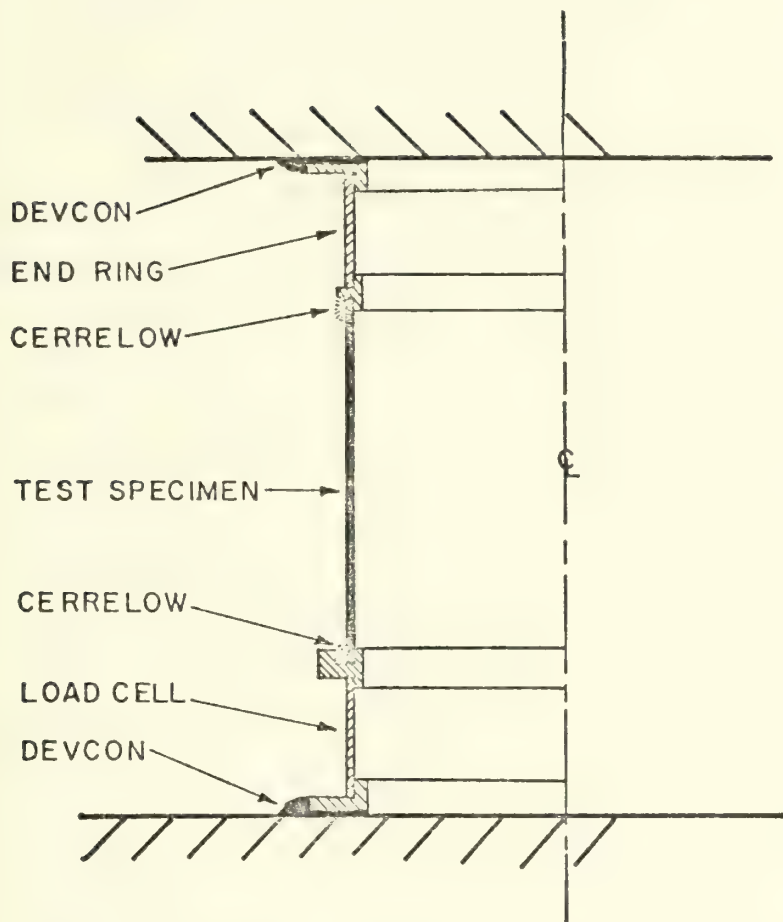
$$W = \phi_s = V = 0 \quad (31)$$

The axial load on the test specimen was applied by means of four screws located at the corners of the square end plates. At each load increment, the screws were adjusted to give as uniform a circumferential load distribution as possible.

In all versions of SATANS, the boundary conditions are represented by the matrix equation

$$\begin{bmatrix} \Omega \end{bmatrix} \begin{bmatrix} N_s \\ N_s \\ Q_s \\ \phi_s \end{bmatrix} + \begin{bmatrix} \Lambda \end{bmatrix} \begin{bmatrix} U \\ V \\ W \\ M_s \end{bmatrix} = \{l\} \quad (32)$$

Boundary conditions can be specified on N_s or U , $N_{s\theta}$ or V , Q_s or W , and ϕ_s or M_s by prescribing input values for the matrices $\begin{bmatrix} \Omega \end{bmatrix}$ and $\begin{bmatrix} \Lambda \end{bmatrix}$. There is one equation (32) for each boundary of the shell. In their present form, SATANS-I



CYLINDRICAL SHELL TESTING CONFIGURATION

FIGURE II

and SATANS-II assume the column matrix $\{l\}$ as zero for all harmonics except the axisymmetric harmonic.

As was mentioned earlier, boundary condition discrepancy between mathematical model and physical experiment is cited as a major contributor to thin shell buckling load disagreement. For this reason, three interpretations of Figure 11 were made. The cylinder was represented as experiencing an axisymmetrically applied force N_s on both ends (condition A), an axisymmetrically applied force N_s on one end and a meridional deflection restraint on U at the other (condition B), and an axisymmetric incremental end shortening U at one end with meridional deflection U restrained at the other (condition C).⁴ These three conditions were investigated on the clamped cylinder, and on a simply supported cylinder for further comparisons and validity checks. Mathematical assignments for boundary conditions A, B and C are shown in Table IV.

C. ANALYSIS

1. Buckling of the Thin Perfectly-Circular Cylinder

A necessary part of this attempt to develop a program for predicting the buckling load of imperfect shells is the establishment of the program's ability to predict the buckling load of perfect shells. Consequently, this section compares the results of a SATANS-II analysis of the thin

⁴It is believed that boundary condition C best represents the actual boundary conditions in Figure 11.

TABLE IV
BOUNDARY CONDITION SPECIFICATIONS

SUPPORT	BOUNDARY CONDITION	BOUNDARY	N_S or U	$N_{S\theta}$ or V	Q_S or W	ϕ_S or M_S
Clamped	A	Upper	$N_S^{(n)} = -N_O$	$V^{(n)} = 0$	$W^{(n)} = 0$	$\phi_S^{(n)} = 0$
		Lower	$N_S^{(n)} = -N_O$			
	B	Upper	$N_S^{(n)} = -N_O$			
		Lower	$U^{(n)} = 0$			
	C	Upper	$U^{(n)} = U_O$			
		Lower	$U^{(n)} = 0$			
Simple	A	Upper	$N_S^{(n)} = -N_O$	$V^{(n)} = 0$	$W^{(n)} = 0$	$M_S^{(n)} = 0$
		Lower	$N_S^{(n)} = -N_O$			
	B	Upper	$N_S^{(n)} = -N_O$			
		Lower	$U^{(n)} = 0$			
	C	Upper	$U^{(n)} = U_O$			
		Lower	$U^{(n)} = 0$			

Note: When $N_S^{(n)}$ or $U^{(n)}$ are specified nonzero, they are set to zero for $n > 0$.

cylinder without imperfections with the bifurcation buckling analysis presented by Almroth [Ref. 147]. Almroth's study deals with the influence of edge constraints on the critical load of cylindrical shells of the type used by Arbocz and Babcock. Equation (33) is given by Almroth for the axisymmetric displacement shape of the simply supported cylinder (boundary condition A) prior to buckling.

$$W = r(2v\gamma\bar{N}_S + \bar{p})(1 + A_1\sin(a_1s)\sinh(a_1s) + A_2\cos(a_2s)\cosh(a_2s)) \quad (33)$$

where

$$a_1 = \kappa(1 + \bar{N}_S)^{\frac{1}{2}}$$

$$a_2 = \kappa(1 - \bar{N}_S)^{\frac{1}{2}}$$

$$\kappa = (2\gamma)^{-\frac{1}{2}}$$

$$\gamma = h/r/(12(1 - \nu^2))^{\frac{1}{2}}$$

$$\bar{N}_S = -N_S/(2\gamma Eh)$$

$$\bar{p} = (pr)/(Eh)$$

$$A_1 = - \left[\frac{(\{1 - \bar{N}_S^2\}^{\frac{1}{2}} \sin(\alpha_1) \sinh(\alpha_2) + \bar{N}_S \cos(\alpha_1) \cosh(\alpha_2))}{(1 - \bar{N}_S^2)^{\frac{1}{2}} (\cosh^2(\alpha_2) + \sin^2(\alpha_2))} \right]$$

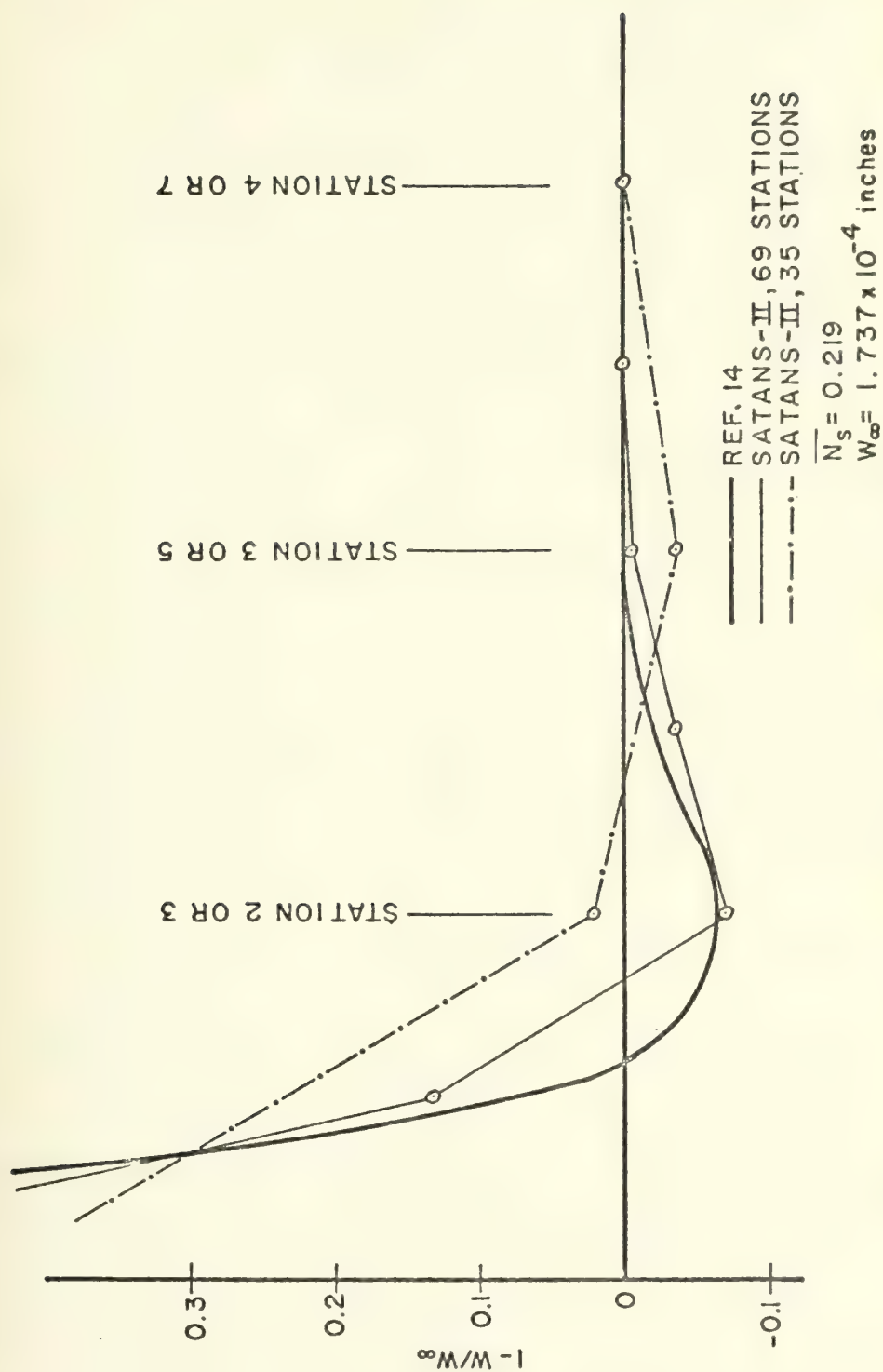
$$A_2 = - \left[\frac{(\{1 - \bar{N}_S^2\}^{\frac{1}{2}} \cos(\alpha_1) \cosh(\alpha_2) - \bar{N}_S \sin(\alpha_1) \sinh(\alpha_2))}{(1 - \bar{N}_S^2)^{\frac{1}{2}} (\cosh^2(\alpha_2) - \sin^2(\alpha_2))} \right]$$

$$\alpha_1 = (a_1 L) / (2r)$$

$$\alpha_2 = (a_2 L) / (2r)$$

and where N_s is the axial load, p is a uniform normal pressure, h is the thickness, E is Young's Modulus, ν is Poisson's ratio, L is the length and r is the radius. This solution is referred to as the consistent prebuckling solution, as opposed to the membrane or classical buckling solution where W is a constant along the meridian. Equation (33) has been solved for shell A-14 and W is plotted as a function of s in Figure 12 for an $\bar{N}_s = 0.21904$. Also shown in Figure 12 is a plot of W versus s obtained from SATANS-II for the same \bar{N}_s for both 35 and 69 meridional station analyses. Note in Figure 12 that even though only a few stations lie in the boundary zone, the trend is towards the correct solution. The displacement of the 35 station solution, however, is only a rough approximation of the actual state.

Almroth has determined the minimum bifurcation buckling load, $N_{s_{cr}}$, and the corresponding value for n for a



MEMBRANE PREBUCKLE DEFLECTION SHAPE, SHELL A-14

FIGURE 12

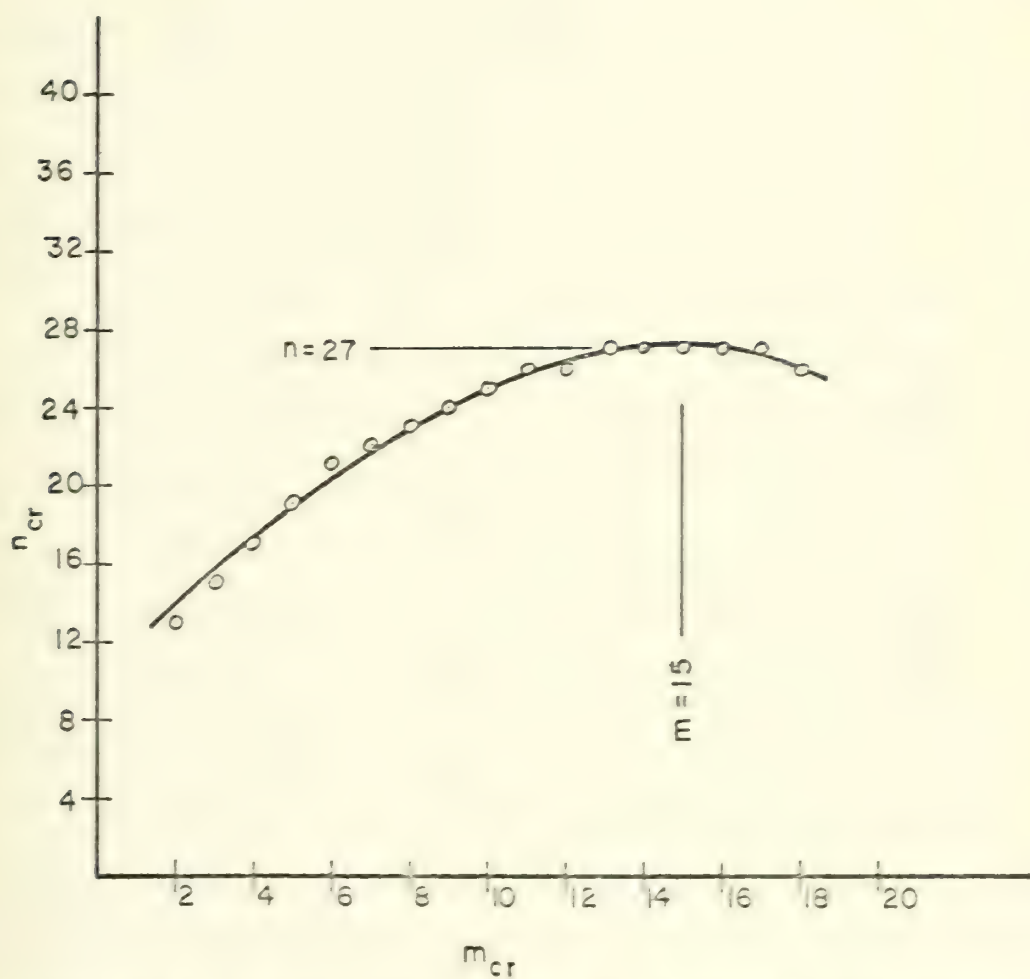
range of parameters L/r and r/h , using both the consistent prebuckling displacement W given by equation (33) and the membrane or classical prebuckling solution, where W is constant, in the Donnell equations. For the simply supported cylinder with an r/h of the same order of magnitude as shell A-14 and an $L/r \approx 0.5$, the critical load is $\bar{N}_{scr} = 0.863$ ($N_{scr} = 39.5$ lb/in) with $n=27$ for the consistent analysis, and $\bar{N}_{scr} = 1.0$ ($N_{scr} = 46$ lb/in) with $n=27$ for the membrane prebuckling analysis. Thus, the consideration of the axisymmetric edge effect, which is analogous to an imperfection, reduces the buckling load by 14.1%. Parametric studies conducted by Almroth show usage of from 100 to 800 meridional stations over the half-length of the shell.

As shown in Ref. 5, use of the membrane prebuckling displacement in Sanders' equations for the simply supported cylinder simplifies the problem to an eigenvalue problem. The buckling mode is

$$W = \delta \sin(m\pi s/L) \cos(n\theta) \quad (34)$$

The equations for the matrix elements in the eigenvalue problem are given in Appendix B. A plot of the critical values of m and n is given in Figure 13. The minimum buckling load occurs when $n=27$ and $m=15$, and corresponds to an N_{scr} of 47.5 lb/in.

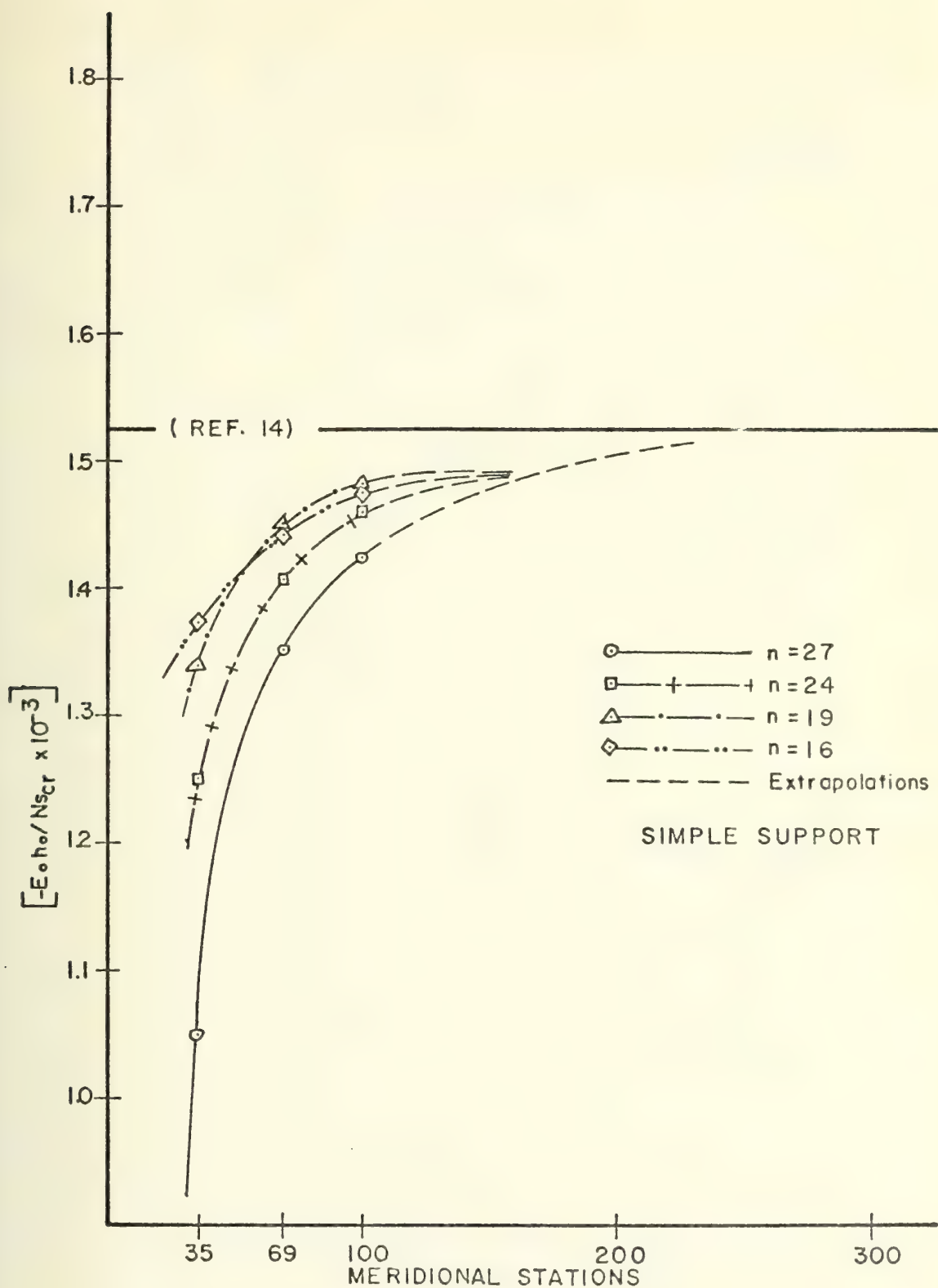
SATANS' ability to predict the minimum bifurcation load of a shell for a specified value of n has been demonstrated by Stilwell [Ref. 15]. SATANS-II analyses of shell



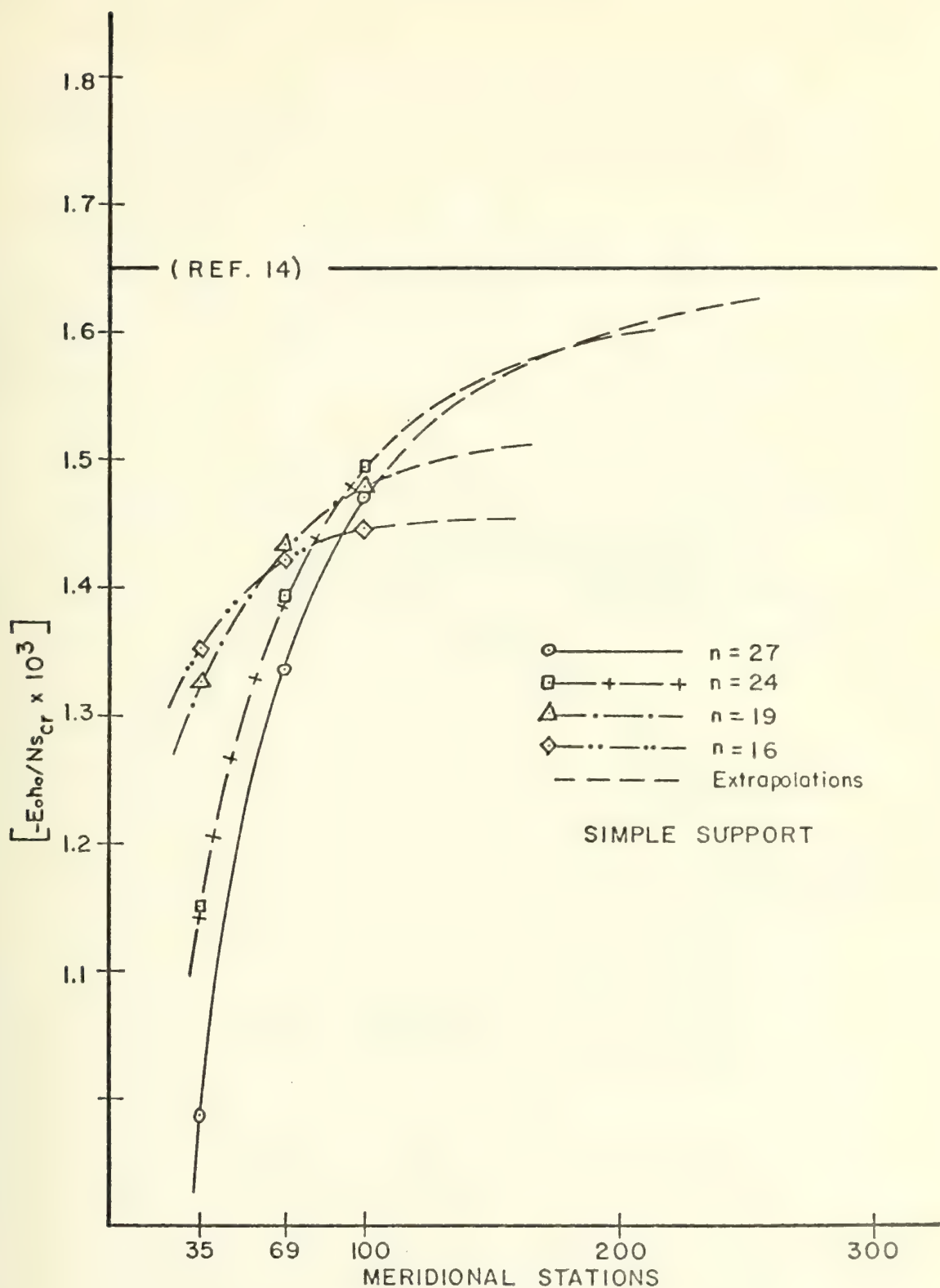
CRITICAL BUCKLING MODES FROM EIGENVALUE
ANALYSIS, SHELL A-14

FIGURE 13

A-14 (simple support, boundary condition A) were conducted over a wide range of n for both the consistent and membrane prebuckle cases. Figures 14 and 15 give plots of the critical buckling load versus the number of meridional stations for various values of n . The dashed parts of the curves are estimated extrapolations. It is clear that as the number of stations is increased, SATANS' critical buckling load decreases and the associated harmonic increases. While the $n=27$ curve predicts the highest buckling load over the range of stations used in both cases, extrapolation of these curves indicates that the $n=27$ curve would predict the lowest buckling load if the number of stations exceeded 200 to 300. Note that the membrane prebuckle critical buckling curves of Figure 15 seem to flatten much more quickly than for the consistent case. This appears to confirm the occurrence of a reduction in critical buckling load due to the presence of an axisymmetric edge restraint, as noted by Almroth. Figures 16 and 17, which present plots of the critical buckling load versus critical buckling mode of shell A-14 for the 35, 69 and 100 station analyses, also predict this lowering of buckling load due to edge restraint. the 'fanning' of curves in Figure 16 indicate an increase in the number of meridional stations will produce a lesser buckling load decrease than in Figure 17. These figures also predict a lower critical buckling load and a higher critical buckling mode with increasing meridional stations.

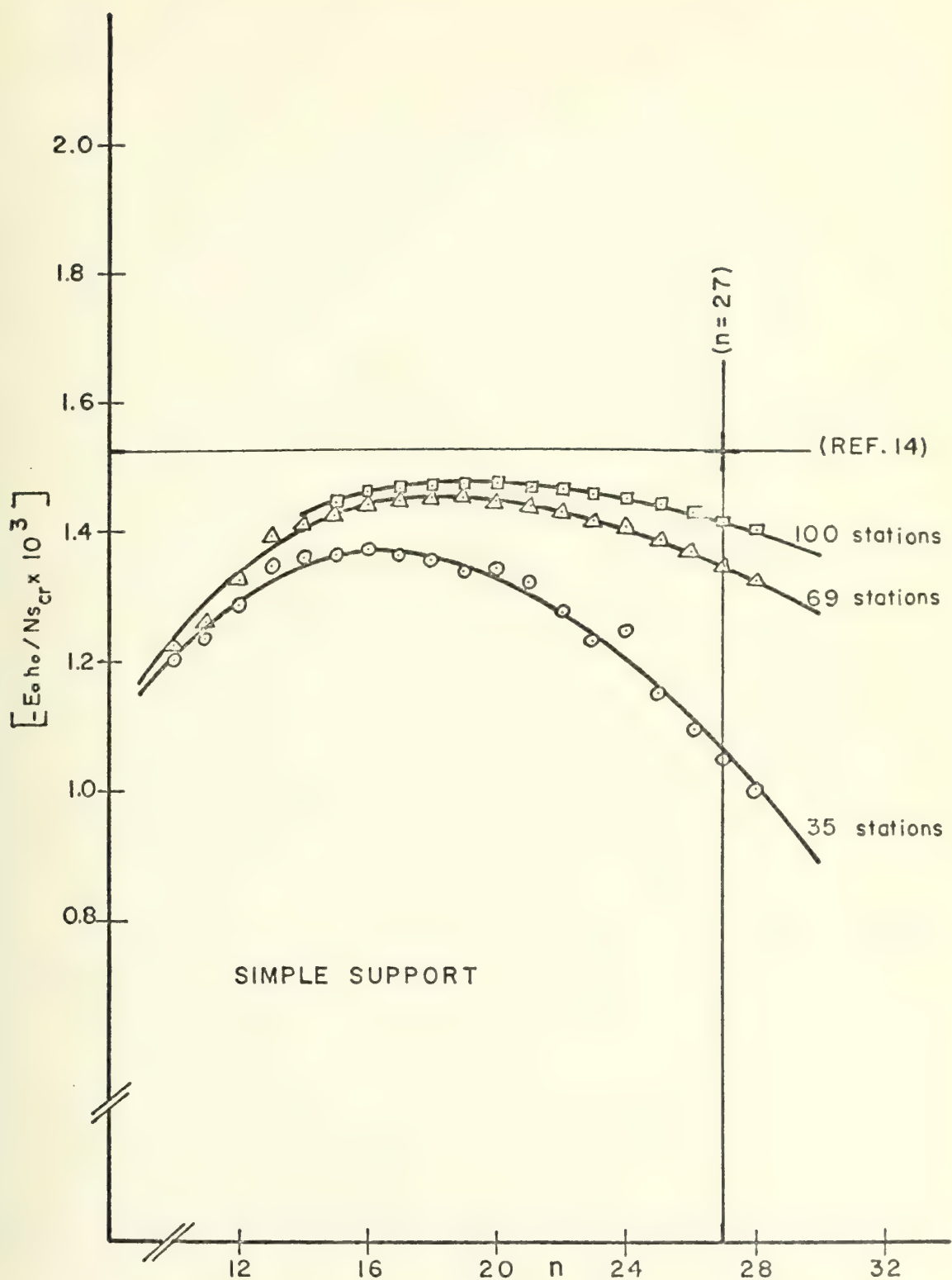


BUCKLING LOAD vs STATIONS
 MEMBRANE PREBUCKLE ANALYSIS, SHELL A-14
 FIGURE 14



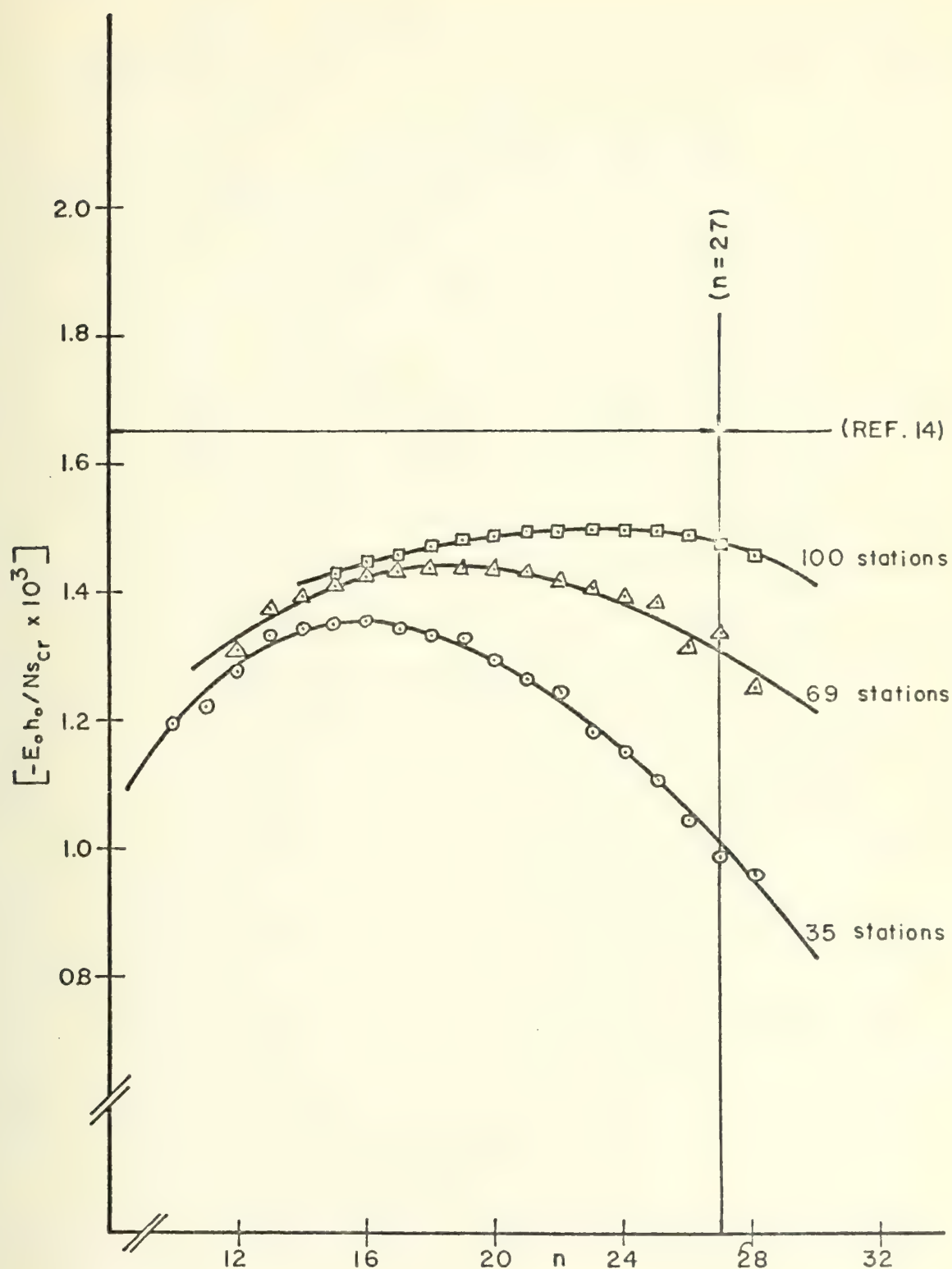
BUCKLING LOAD vs STATIONS
CONSISTENT ANALYSIS, SHELL A-14

FIGURE 15



BUCKLING LOAD vs BUCKLING MODE
MEMBRANE PREBUCKLE ANALYSIS, SHELL A-14

FIGURE 16



BUCKLING LOAD vs BUCKLING MODE
CONSISTENT ANALYSIS, SHELL A-14

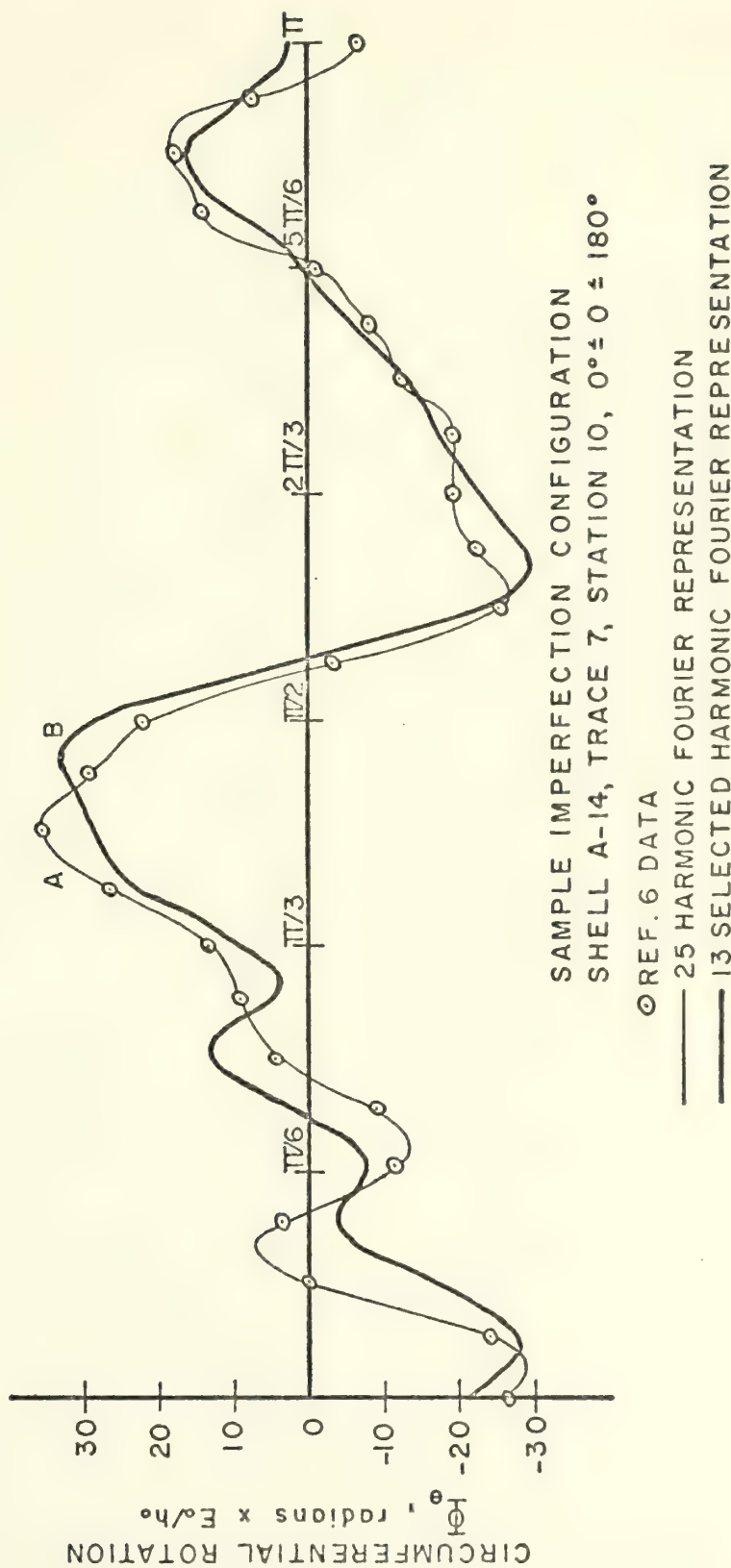
FIGURE 17

Figures 14 through 17 indicate that, given enough stations, SATANS-II can predict a critical buckling load and mode for the thin-walled shell that agrees with Almroth's prediction. However, for the 35-station analysis, which is the analysis to be used for the imperfect shells, the critical load of 51.1 lb/in and harmonic of 16 are significantly different from the correct values. As a result of the coarse mesh used by SATANS-II in the following analysis, the critical buckling loads for the imperfect shells will be higher than the experimental values. Nevertheless, it will be shown that the introduction of a totally arbitrary, naturally-occurring imperfection pattern on the imperfection-free mathematical model of shell A-14 results in a significant reduction in critical buckling load. This is not only sufficient to demonstrate the capability of the program to perform the analyses, but it also provides an estimate of the degree of imperfection sensitivity of the shell. It will be shown that application of this estimate of sensitivity to Almroth's buckling load provides an accurate prediction of the buckling load of the imperfect shell.

2. Introduction of Imperfections

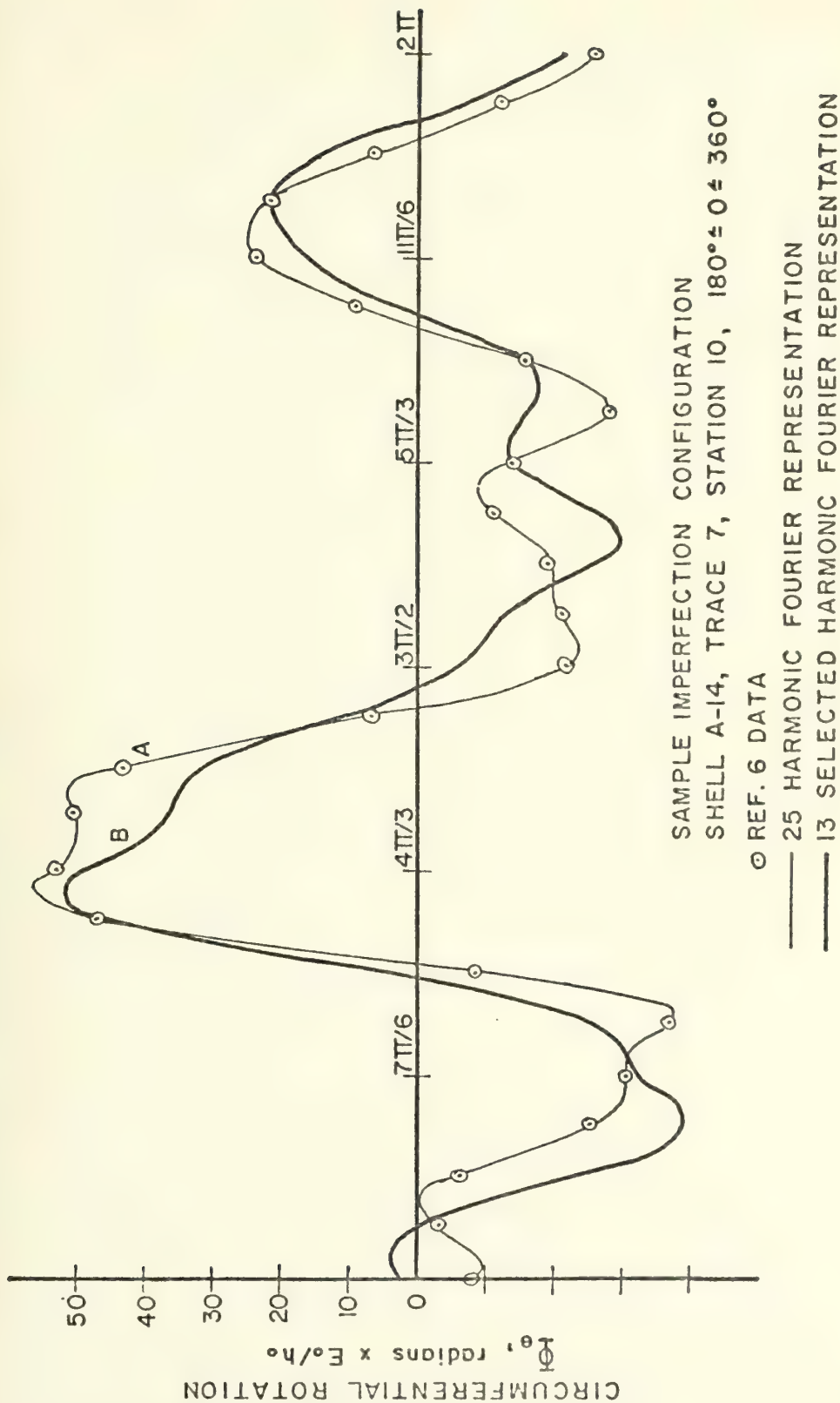
When prepared as indicated in section V.A.2, the imperfection rotation Fourier coefficients are in a form for direct introduction into SATANS-II. Figure 18 shows the applicability of the periodogram method of expansion of the imperfection data according to equations (29) and (30).





CIRCUMFERENTIAL IMPERFECTION ROTATION REPRESENTATIONS - Φ_θ vs θ

FIGURE 18 - PLATE 1 OF 2



CIRCUMFERENTIAL IMPERFECTION ROTATION REPRESENTATION - Φ_θ vs θ

FIGURE 18 PLATE 2 OF 2

Curve A is the graph of a solution of equation (30) representing the circumferential imperfection rotation at station 10 plotted against θ . It is seen to pass smoothly through all the data points.

Only the zeroeth harmonic ($n=0$) and twelve others can be accommodated by SATANS-II. In the case where the shell response and the imperfection are limited to a few major harmonics, all important harmonics can be included in the analysis. For example, the $\lambda=6$ shallow spherical cap has only the $n=2$ harmonic as the major harmonic of response, and the imperfection curves of Figure 8 require only a few Fourier harmonics for adequately accurate representation. Also, the few required harmonics for imperfection representation are in the vicinity of the $n=2$ harmonic. This problem is well within the limits of SATANS-II.

The thin-walled cylinder, however, responds in a large number of harmonics. This can be seen in Figure 16 -- many harmonics produce critical buckling loads close to the value of the minimum critical buckling load. Additionally, the imperfection requires many harmonics. Thus, as mentioned in section V.A.4, a difficult selection of harmonics for use in the analysis must be made. Six harmonics were selected on the basis of the shell response and six harmonics were selected on the basis of the imperfection characteristics. Of course, all 13 selected harmonics will contain both the imperfection characteristics and the shell response characteristics.

For the shell response harmonic selection, the critical buckling loads of shell A-14 for specified values of n , discussed in the previous section, are presented in Table V. Values were assigned to each harmonic according to the legend of Table V and six were selected based on these assignments.

For the imperfection harmonic selection, the sums of the absolute values of the Fourier coefficients for both sine and cosine terms of both meridional and circumferential imperfection rotations were tabulated and are presented in Table VI. Values were again assigned to each harmonic according to the legend of Table V, and six additional harmonics were selected.

The complete harmonic selection process for shell A-14 was accomplished by a comparison of the values assigned to each harmonic for both the imperfection and the shell. Table VII presents this comparison along with the actual harmonics selected for the analysis. Note that no overlap of important harmonics occurred, as the imperfection required harmonics seven and below and the shell required harmonics ten and above. The harmonics used with shell A-13 were $n=0$ and $n=2,3,4,6,8,9$ for the imperfection and $n=10,11,12,13,14,15$ for the shell response with a simple support, and $n=13,14,15,16,17,18$ for the shell response with a fixed support.

Once the specific harmonics for inclusion in the analysis are determined, the appropriate $\bar{\Phi}_S$ and $\bar{\Phi}_\theta$

TABLE V
CRITICAL BUCKLING LOADS OF INDIVIDUAL HARMONICS

HARMONIC	FIXED SUPPORT			SIMPLE SUPPORT		
	BUCKLING LOAD $\times P_0$	P/P_{cr}	ASSIGNED VALUE	BUCKLING LOAD $\times P_0$	P/P_{cr}	ASSIGNED VALUE
2	6+			6+		
3	6+			6+		
4	5.320			6+		
5	5.120			5.720		
6	4.250			4.080		
7	2.415	2.380		2.445	2.390	
8	1.611	1.550		1.661	1.630	
9	1.221	1.180	-	1.291	1.260	
10	1.093	1.050	*	1.157	1.130	-
11	1.088	1.050	*	1.131	1.110	-
12	1.069	1.030	*	1.085	1.060	+
13	1.038	1.000	**	1.037	1.014	*
14	1.043	1.010	*	1.029	1.006	*
15	1.054	1.020	*	1.026	1.004	*
16	1.058	1.020	*	1.022	1.000	**
17	1.074	1.030	*	1.030	1.008	*
18	1.083	1.040	*	1.037	1.015	*
19	1.120	1.080	+	1.040	1.018	*
20	1.200	1.160	-	1.069	1.050	+
21	1.240	1.190	-	1.090	1.070	+
22	1.320	1.270		1.130	1.110	-
23	1.340	1.290		1.168	1.144	-
24	1.344	1.290		1.200	1.170	-

Notes: 1. $P_0 = 50$ psi.

2. The trigger in each harmonic is an internal pressure of 10^{-7} psi.

3. Each analysis included the indicated harmonic plus $n=0$.

4. Value assignments are according to the following legend:

SYMBOL	VALUE
**	absolutely necessary
*	great
+	moderate
-	slight
	none

TABLE VI

RELATIVE IMPERFECTION ROTATION COEFFICIENT SIZES, SHELL A-14

HARMONIC	Φ_s				Φ_θ			
	COS	VALUE	SIN	VALUE	COS	VALUE	SIN	VALUE
0	5.32	*	0.00		0.01		0.00	
1	3.18	*	2.03	*	0.34		0.40	
2	4.85	*	9.08	*	9.46	*	13.26	*
3	1.50	+	7.34	*	13.46	*	3.11	+
4	2.80	*	1.74	+	6.28	+	28.54	*
5	1.71	+	0.79		7.84	+	9.98	*
6	1.33	+	1.48	+	5.20	+	4.07	+
7	0.56		1.85	+	11.20	*	6.26	+
8	0.57		0.86	-	7.88	+	1.92	-
9	0.46		0.69		3.75	-	2.88	-
10	1.10	+	0.85	-	5.60	+	5.99	-
11	0.77		0.91	-	2.17	-	1.68	-
12	1.39	+	1.67	+	5.73	+	6.77	+
13	1.05	+	1.01	+	2.98	-	3.38	+
14	0.77		0.83	-	2.60	-	1.55	+
15	0.91	-	0.46		1.05	-	1.26	-
16	1.09	+	0.69		0.73		2.17	-
17	0.54		1.15	+	1.31	-	1.17	-
18	0.83	-	1.03	+	1.16	-	1.46	-
19	0.94	-	0.78		0.48		1.04	-
20	0.41		0.81	-	0.49		0.48	
21	0.42		0.68		0.45		0.19	
22	0.43		0.63		0.31		0.15	
23	0.68		0.58		0.11		0.10	
24	0.32		0.01		0.01		0.01	

- Notes: 1. Numbers represent the sums of the absolute values of the imperfection rotation coefficients over the entire shell.
2. Value assignments were made according to the legend of Table V.

TABLE VII
HARMONIC SELECTION, SHELL A-14

H A R M O N I C	$\bar{\phi}_s$ COEFFI- CIENT		$\bar{\phi}_\theta$ COEFFI- CIENT		SHELL RESPONSE		SELECTED HARMONICS		
	COS	SIN	COS	SIN	FIXED SUPPORT	SIMPLE SUPPORT	FIXED SUPPORT		SIMPLE SUPPORT
							PRIMARY	ALTER- NATE	
0	*						X	X	X
1	*	*					X	X	X
2	*	*	*	*			X	X	X
3	+	*	*	+			X	X	X
4	*	+	+	*			X	X	X
5	+		+	*			X	X	X
6	+	+	+	+					
7		+	*	+			X	X	X
8		-	+	-					
9			-	-	-				
10	+	-	+	+	*	-	X		
11		-	-	-	*	-	X		
12	+	+	+	+	*	+	X	X	
13	+	+	-	+	**	*	X	X	X
14		-	-	-	*	*	X	X	X
15	-		-	-	*	*	X	X	X
16	+			-	*	**		X	X
17		+	-	-	*	*		X	X
18	-	+	-	-	*	*			X
19	-			-	+	*			
20		-			-	+			
21						+			
22						-			
23						-			
24						-			

- Notes: 1. Harmonic zero is required by SATANS-II.
2. Value assignments are taken from Tables V and VI.
3. 'X' indicates a selected harmonic.

coefficients (obtained from the periodogram) must be loaded into the SATANS-II arrays PHIXB and PHITB (performed in subroutine IMPERF) in the following order:

1. The first KMAX locations contain the $\bar{\phi}_s$'s (or $\bar{\phi}_\theta$'s) of mode 0.
2. The second KMAX locations contain the $\bar{\phi}_s$'s (or $\bar{\phi}_\theta$'s) of mode -i.
3. The third KMAX locations contain the $\bar{\phi}_s$'s (or $\bar{\phi}_\theta$'s) of mode +i.
4. The fourth KMAX locations contain the $\bar{\phi}_s$'s (or $\bar{\phi}_\theta$'s) of mode -j.
5. Et cetera, until all modes are entered.

KMAX is the FORTRAN name for the number of meridional stations to be included in the analysis. One special case must be considered where the shell has an initial or a final pole. In this case, rotations are zero in all but the n=1 harmonic. If this harmonic is included in the analysis, the imperfection rotation coefficients at the first (or last) station should all be set to zero except for the coefficients in the n=1 harmonic. In the case of an initial pole, the coefficient associated with the -1 index should be placed in the first location of the second set of KMAX locations in arrays PHIXB and PHITB, and the coefficient associated with the +1 index should be placed in the first location of the third set of KMAX locations. For a final pole, the coefficient associated with the -1 index should be placed in the first location of the next to last set of KMAX

locations, and the coefficient associated with the +1 index should be placed in the first location of the last set of KMAX locations.

To illustrate the effect of the selection of harmonics on the imperfections, equation (30) was solved using only the 13 selected harmonics. The resulting circumferential imperfection rotation representation at station 10 appears as curve B in Figure 18. To determine the effect of the inability to include all nine harmonics deemed important to the shell response with a fixed support, an alternate selection of harmonics was made for this case (see Table VII). A complete SATANS-II analysis was performed with this alternate selection and the results were found to differ from those using the primary harmonic selection by approximately 2%.

3. Results

Six analyses (boundary conditions A, B and C with clamped and simple supports) were made for each shell where the imperfections were reduced by a factor of 1000. These 'minute' imperfections acted only as triggers for the modal deflections and the resulting solutions gave the shell's consistent critical bifurcation buckling loads. These results are presented in Table VIII under the category of minute imperfections. Selected cases were analyzed where the minute imperfection triggers were replaced by a 10^{-7} psi internal pressure trigger and the results were identical.

TABLE VIII

BUCKLING LOADS, MINUTE VERSUS FULL IMPERFECTIONS

SUPPORT	DEGREE OF IMPERFECTION	BOUNDARY CONDITION	SHELL A-14		SHELL A-13	
			BUCKLING LOAD (lb/in)	REDUCTION (%)	BUCKLING LOAD (lb/in)	REDUCTION (%)
CLAMPED	Minute	A	47.20		35.60	
		B	51.90		50.60	
		C	56.00		55.10	
	Full	A	43.00	8.90	*	-
		B	46.60	10.20	40.70	19.55
		C	49.00	12.50	40.15	22.70
SIMPLE	Minute	A	48.00		46.60	
		B	51.10		50.60	
		C	54.10		55.30	
	Full	A	42.80	10.8	34.30	26.30
		B	46.10	9.8	38.00	24.90
		C	47.90	11.5	42.7	22.80

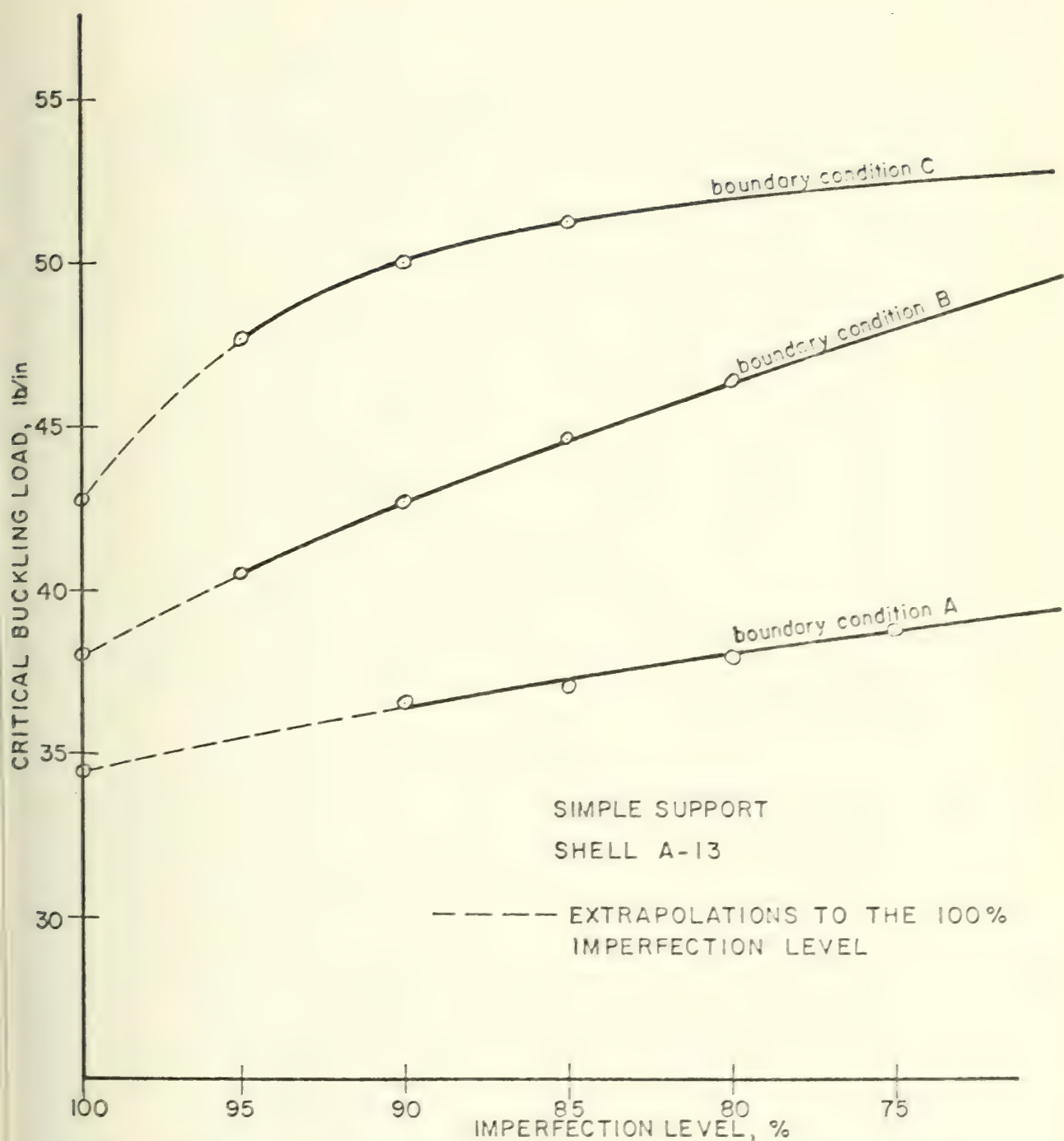
* insufficient data.

The same six analyses for each shell were also made using the full imperfections. Results for the final load at which a converged solution could be obtained are also shown in Table VIII along with the percent reduction in the buckling load for each case.

Since the full imperfection is introduced at the first load step as a pseudo load, the program requires considerably more iterations than normal to obtain the first

solution. The subsequent solution usually proceeds in a completely normal fashion. However, convergence difficulties may be encountered in cases where the imperfection level is high. This was the case with shell A-13, as some of its imperfection deviations exceeded $3\frac{1}{2}$ shell wall thicknesses, and fluctuated over 6 thicknesses in a θ arc of only 53° . For shell A-13, a 70% initial imperfection level was the maximum for which a converged solution could be obtained on the first load step. Consequently, buckling loads were computed with a maximum of 80%, 85%, 90% and 95% of the full imperfection data. For the 80% imperfection, 70% was introduced at load step one and 10% was added at load step two. For the three other solutions, 5% increments were added at various load steps. Since SATANS uses the total value of the variables and not the incremental values, each intermediate solution is correct for its given load and imperfection level, but the load-deflection characteristics are invalid. An imperfection level of 95% was the absolute maximum for which a converged solution could be obtained for for shell A-13, and a short 5% extrapolation was made to obtain the data presented in Table VIII for this shell.⁵ This procedure is illustrated for the shell with a simple support in Figure 19. A comparison of load steps and iterations for both shells under these conditions is presented in Table IX.

⁵For shell A-13 with clamped boundary condition A, a converged solution was not obtained for an imperfection level above 85%. This condition is therefore not included in the results. A similar problem was encountered by Dantone /Ref. 57 with this boundary condition.



CRITICAL BUCKLING LOAD vs IMPERFECTION LEVEL

FIGURE 19

TABLE IX
LOAD STEPS VERSUS ITERATIONS

SHELL A-14			SHELL A-13		
LOAD STEP	INCREMENTAL IMPERFECTION ADDITION	ITERATIONS REQUIRED	LOAD STEP	INCREMENTAL IMPERFECTION ADDITION	ITERATIONS REQUIRED
1	100%	7	1	70%	24
2		2	2	10%	6
3		2	3	5%	16
4		3	4		12
5		3	5		9
6		3	6		3
7		3	7		8
8		3	8		6
9		3	9	5%	19
10		3	10		3
11		3	11	5%	60
12		4	12		58
13		4	13		18
14		4	14		16
15		5	15		16
16		7	16		12
17		10	1 st load step size reduction		
18		16	17		4
1 st load step size reduction			2 nd load step size reduction		
19		3	18		3
20		2	19		2
2 nd load step size reduction			3 rd load step size reduction		
21		3	termination		
3 rd load step size reduction			(P _{Cr} =40.6 lb/in)		
22		2			
termination					
(P _{Cr} =46.0 lb/in)					

Note: This data corresponds to boundary condition B, simple support.

D. DISCUSSION

The SATANS-II investigation of the buckling load of the perfect cylinder (discussed in section V.C.1) indicates that a numerical analysis using 35 stations results in a higher predicted buckling load than the correct analytical buckling load. Consequently, the SATANS-II results for the prediction of the critical buckling load of the imperfect cylinder are expected to be high. Table X compares the results of the SATANS-II analysis (column A) with the actual experimental results of Ref. 6 (column E) and shows this to be the case. The use of more stations to lower the predicted buckling load can be accomplished only by either decreasing the number of harmonics considered in the analysis or by increasing overall program size. Both of these choices are impractical at the present time since the number of harmonics already in the analysis appears to be low considering the number required for shell response and the imperfection data, and the present program core storage requirement is large.

It is possible, however, to apply the SATANS-II results for the percent reduction in the buckling load due to the imperfections (see Table VIII and Column B, Table X) to a more accurate buckling load of the perfect shell from an analysis like that of Almroth to get a more accurate estimate of the buckling load of the imperfect shell. When the percent reduction from the consistent minute imperfection buckling load to the consistent full imperfection buckling

load, determined by SATANS-II, is applied to the accurate consistent bifurcation buckling load of Almroth (column C, Table X), the result is a prediction of the actual buckling load of the imperfect shell. These imperfect shell predicted buckling loads are shown in column D of Table X and are seen to be very close to the actual buckling loads of the imperfect shells, given in Column E of Table X.

The percent reductions in buckling loads determined by SATANS-II are considered to be applicable to the accurate consistent bifurcation buckling loads of Almroth since the imperfection characteristics are seen to be concentrated in the lower ten harmonics (see Table VI), where shell response does not appear to be largely affected by the number of meridional stations in the analysis (see Figures 14 and 15).

E. PROPOSED ANALYSIS PROCEDURE

When the SATANS-II predicted buckling load of the perfect shell is higher than the analytical prediction, as is the case with the extremely thin-walled cylinder, the following procedure is proposed.

1. Determine the consistent bifurcation buckling load as close as possible.
2. Determine the buckling load with SATANS-II using minute imperfections.
3. Determine the buckling load with SATANS-II using full imperfections and compute the percent reduction from step 2.
4. Apply this reduction to the buckling load of step 1.

TABLE X
PREDICTED BUCKLING LOADS OF THE IMPERFECT SHELL

	BOUNDARY CONDITION	A SATANS-II PREDICTIONS WITH FULL IMPERFECTIONS (lb/in)	B SATANS-II PERCENT REDUCTION (%)	C ALMROTH'S ANALYTICAL CONSISTENT PREDICTION /Ref. 14/ (lb/in)	D PERCENT REDUCTION APPLIED TO ANALYTICAL PREDICTION (lb/in)	E ACTUAL EXPERIMENTAL RESULTS /Ref. 6/ (lb/in)
SHELL A-14 CLAMPED	A	43.00	8.9	42.5	38.70	35.4
	C	49.00	12.5	42.0	36.75	
SHELL A-13 CLAMPED	C	40.15	27.2	42.0	30.50	31.4

This procedure is expected to work only when the result of step 2 is reasonably close to that of step 1.

VI. CONCLUSIONS

A. SATANS-I

The computer program SATANS-I has been constructed from SATANS. It is a more user-oriented version of the program by Ball, whose primary modification is the introduction of an equipment-independent graphical output ability.

B. SATANS-II

SATANS-II is a further modification of SATANS where the restriction that all applied loading and initial conditions be symmetrical about some meridional plane has been removed. In addition, SATANS-II possesses the capability of including experimentally-determined naturally-occurring imperfections or malformations induced during fabrication, mathematically described imperfections of any type, frequency or location, or graphically described imperfections due to damage suffered after fabrication. The imperfections are introduced into SATANS-II by using the imperfection deviations along discrete circumferences to determine the associated meridional and circumferential imperfection rotations at the mesh points, performing a Fourier expansion on these rotations and then use of the Fourier coefficients. In cases where the important shell response harmonics and the harmonics required for imperfection representation exceed the limits of SATANS-II, a selection must be made.

Instructions for the use of both SATANS-I and SATANS-II are contained in Appendix C and complete listings of both programs are given in other appendices. SATANS-I requires 210 k-bytes (53k words) of core on an IBM-360/67 computer using 'single precision,' while SATANS-II requires 426 k-bytes (107k words) of core. Imperfect shell analyses performed by SATANS-II in this report used a 35 station meridian and 25 circumferential modes. Execution times were an average of 60 minutes, although execution times of between 15 and 30 minutes can be expected for analyses using 15 or less modes.

C. IMPERFECT SHELLS

SATANS-II has demonstrated an ability to perform a numerical structural analysis on the arbitrarily imperfect shell of revolution. Imperfections of over three shell wall thicknesses in depth have been handled. Increasing solution difficulty relates directly to increasing imperfection complexity and increasing shell instability. In those cases where extremely thin-walled shells result in perfect (imperfection-free) buckling loads from SATANS that are higher than analytically determined buckling loads, a procedure has been devised where the percent reduction due to the introduction of full imperfections can be applied to the analytically determined buckling load, and an imperfect shell buckling load predicted. These predictions have been shown to be very close to the actual imperfect shell buckling loads for the two specific axially loaded cylinders considered.

ADDENDUM

A DIGITAL COMPUTER STUDY OF THE EFFECTS OF VARIOUS ARTIFICIAL BOUNDARY CONDITIONS AT THE POLE OF A SHALLOW SPHERICAL SHELL

I. INTRODUCTION

The design of many shell of revolution structures requires continuity of the structure at one or both poles. At such a pole, the radius becomes zero and special conditions must be imposed on the deflections and the stress resultants to assure finite strains and equilibrium. As a consequence of the difficulty of incorporating these conditions into a structural analysis computer program, many researchers model such a structure as having a small hole or a small rigid plug at the pole. This addendum investigates the applicability of the various artificial pole conditions by examining their effect on the stability of a clamped spherical cap subjected to axisymmetric and asymmetric pressure loads.

Various polar artificialities are discussed and the boundary equations associated with their use are given, as are those for the continuous pole. A determination of the applicability of the various poles is made by comparisons of their effect on the shell buckling loads and by comparing the errors in the normal displacement and meridional membrane force distributions incurred by their use.

The computer program SATANS, developed by Ball [Ref. 37], was used to conduct this study and selected results of the SATANS analyses are checked for validity with the computer programs BOSOR [Ref. 157] and STAGS [Ref. 157]. SATANS has a special routine that incorporates the proper conditions to be enforced at a pole. BOSOR also has this special ability but STAGS is not yet so equipped.⁶

SATANS, a one-dimensional finite difference equilibrium formulation based on Sanders' equations, was run on an IBM-360/67 computer under OS/MVT. BOSOR, a one-dimensional finite difference energy formulation based on the Donnell equations, was run on a UNIVAC-1108 computer under EXEC-VIII, while STAGS, a two-dimensional finite difference energy formulation was run on a CDC-6600 computer under COMPASS-12.

⁶As of February, 1972.

II. STATEMENT OF THE PROBLEM

A. GEOMETRY AND LOADING

The structure investigated is the clamped shallow spherical cap whose geometry can be specified by the single nondimensional parameter λ , where

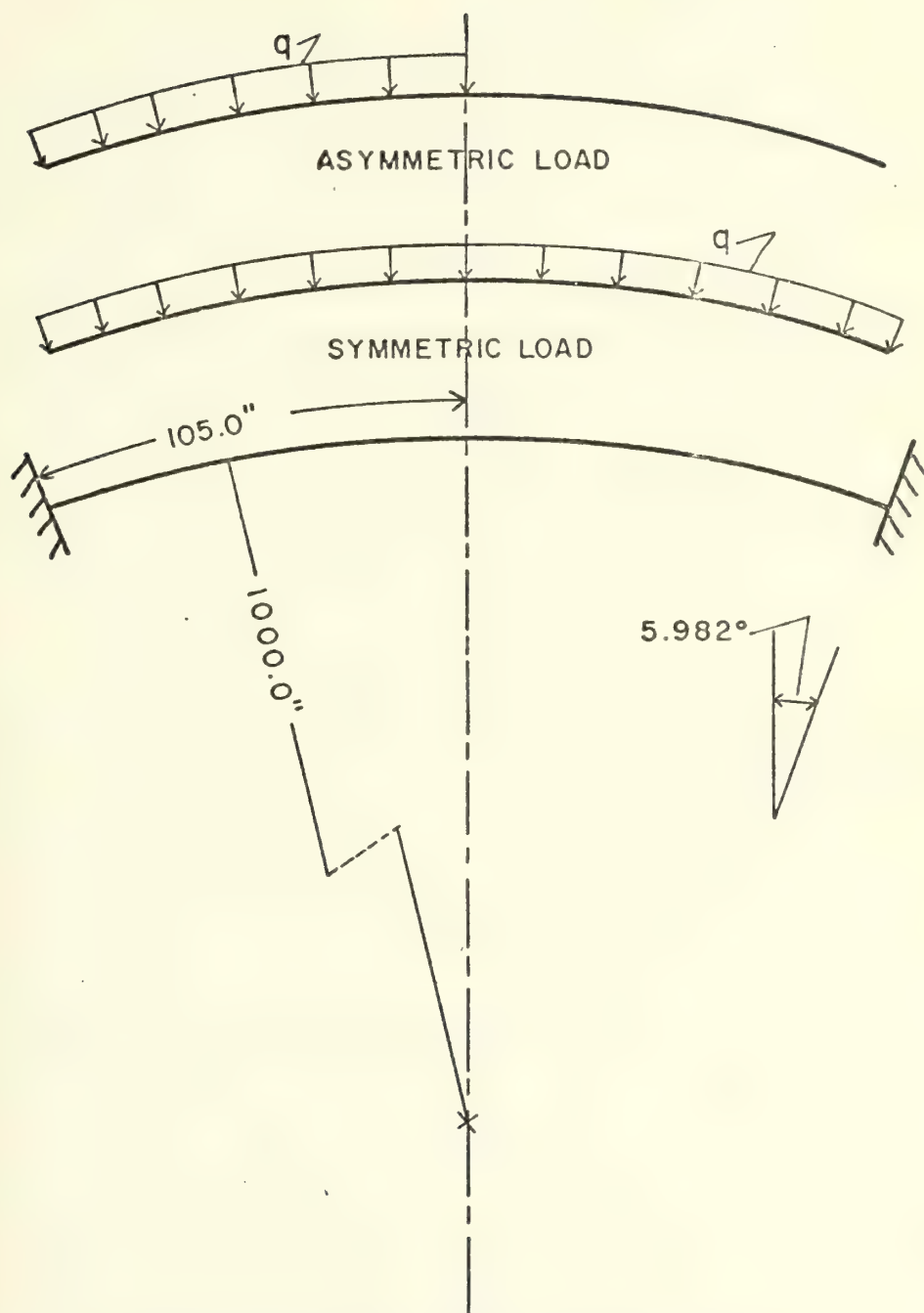
$$\lambda = 2(3(1-\nu^2))^{\frac{1}{4}}(H/h)^{\frac{1}{2}} \quad (\text{AD-1})$$

and where H is the rise of the shell and h is its thickness. For this study, Poisson's ratio was taken as 0.3, Young's modulus as 30×10^6 psi and h as 1.0 inches. Shells of $\lambda = 6$ and $\lambda = 18$ were investigated however only the results of the $\lambda = 6$ shell are given in detail.

The classical buckling pressure of a complete sphere is given as q_0 , where

$$q_0 = 2Eh^2/R_s / (3(1-\nu^2))^{\frac{1}{2}} \quad (\text{AD-2})$$

and where R_s is the radius of curvature. For this study, two loading conditions were used. The cap was subjected to a uniform normal pressure (symmetrical load) and a uniform normal pressure over half the cap (asymmetrical load). These loadings are shown in Figure 20. In each case, the magnitude of the applied pressure is q and the buckling results are presented as the ratio q/q_0 .



$\lambda = 6$ SHALLOW SPHERICAL CAP
GEOMETRY AND LOADING

FIGURE 20

B. POLE CONDITIONS

A shell is considered to have a pole if the structure is continuous through its axis of revolution. At a pole, the radius is zero and special conditions must be imposed on deflections u , v and w , and on the meridional moment m_s to assure finite strains and equilibrium. The computer program SATANS is equipped with the capability to impose these special boundary conditions. A full derivation of the required conditions is given in Appendix D of Ref. 3. The results of that derivation are presented below.

For the zeroeth Fourier harmonic ($n=0$):

$$\begin{aligned}u &= v = m_{s\theta} = t_{s\theta} = 0 \\w' &= m_s' = 0 \\m_{s\theta} &= m_s \\t_\theta &= t_s = b(1-\nu) (u' + \omega_s w + \frac{1}{2}\beta_s) - t_T\end{aligned}\tag{AD-3}$$

For the first Fourier harmonic ($n=1$):

$$\begin{aligned}(u \pm v) &= w = u' = 0 \\m_s &= m_\theta = m_{s\theta} = 0 \\t_s &= t_\theta = t_{s\theta} = 0\end{aligned}\tag{AD-4}$$

When a double sign (\pm) appears, the upper sign applies for an initial pole and the lower sign applies for a final pole.

For the second harmonic (n=2):

$$\begin{aligned}
 u &= v = w = m_s' = 0 \\
 t_\theta &= -t_s = \pm t_{s\theta} \\
 m_\theta &= -m_s = \pm m_{s\theta} \\
 t_s &= b(1-v) (u' + \frac{1}{2}\beta_s)
 \end{aligned}
 \tag{AD-5}$$

For all higher harmonics (n>2):

$$\begin{aligned}
 u &= v = w = 0 \\
 m_s &= m_\theta = m_{s\theta} = 0 \\
 t_s &= t_\theta = t_{s\theta} = 0
 \end{aligned}
 \tag{AD-6}$$

For rotations at the pole, for the first harmonic (n=1):

$$\begin{aligned}
 \phi_s &= -w' + \omega_s u \\
 \phi_\theta &= \mp \phi_s \\
 \phi &= \frac{1}{2} (2v' \pm u')
 \end{aligned}
 \tag{AD-7}$$

and for all other harmonics,

$$\begin{aligned}
 \phi_s &= \phi_\theta = 0 \\
 \phi &= \frac{1}{2} (2v' \pm nu')
 \end{aligned}
 \tag{AD-8}$$

Numerous artificial boundary conditions can be substituted for the correct conditions at the pole. For example,

a small hole can be placed at the pole and can exist with the boundary completely free, or have conditions imposed such that the meridional rotation ϕ_s and deflection u are restrained. A meridional line load N_s of magnitude sufficient to replace that 'lost' due to the hole can be applied. Another artificial pole condition is that best described as a small rigid plug. Here, both the meridional as well as the circumferential deflection is restrained along with the meridional rotation. A normal line load $-P$ whose total equals that loading 'lost' due to the hole or plug can be applied at the boundary of the hole or plug.

Preliminary numerical studies showed that the line loadings effected a noticeable change in the solution only in the immediate area of the boundary, hence, the line loads were not considered further. Preliminary studies also showed that the cap behavior exhibited only minor changes when the small hole with a partially restrained edge was substituted for that with a totally free edge. This investigation, therefore, concentrated on two artificial boundary conditions at a pole and on the continuous pole. The artificial pole conditions used and their associated mathematical expressions are:

1. The totally free hole:

$$N_s = 0 ; N_{s\theta} = 0 ; Q_s = 0 ; M_s = 0 \quad (\text{AD-9})$$

2. The rigid plug:

$$U = 0 ; V = 0 ; Q_s = 0 ; \phi_s = 0 \quad (\text{AD-10})$$

C. TYPES OF ANALYSIS

Three types of buckling analyses were conducted on the cap under each of the three pole conditions and these were:

1. The nonlinear axisymmetric analysis of the symmetrically loaded clamped spherical cap.
2. The bifurcation buckling analysis of the symmetrically loaded clamped spherical cap with nonlinear prebuckling axisymmetric displacements.
3. The nonlinear collapse analysis of the asymmetrically loaded clamped spherical cap.

In the artificial pole cases, hole and plug radii were taken as 0.17%, 1.70% and 3.30% of the shell meridian. The SATANS analyses used a 100 station meridian for the axisymmetric cases, a 66 station meridian with three harmonics for the bifurcation buckling cases and a 40 station meridian with five harmonics for the nonsymmetric collapse cases.

III. DISCUSSION

The results of the SATANS analyses are presented in Table XI for all cases. Also presented are the normal deflections of a point located at 47.5% of the meridian which is near the maximum displacement for each case.

A. THE $\lambda = 6$ SHALLOW CAP

1. The Axisymmetric Analysis

In Table XI, the buckling load is seen to be basically unaffected by the altered pole conditions. The normal displacement, however, is seen to show a marked sensitivity to the pole conditions. Figure 21 shows this sensitivity. In the figure, the upper curve is the deflection shape of the cap at incipient buckling with a continuous pole, and the lower curves represent the percent error induced in the normal displacement at incipient buckling by the introduction of artificial poles. The 0.17% plug and the 0.17% hole behaved identically. Holes, in general, produced an average 20% error in the normal displacement and this error existed over the entire shell. The plugs, preserving the continuity of the structure, produced an average 10% error, but again, this error extended over the entire shell. Figure 22 shows the behavior of the meridional force N_s for the continuous pole case, and the N_s error due to the artificial pole conditions, also at incipient buckling. Large errors existed in all cases over the first third of

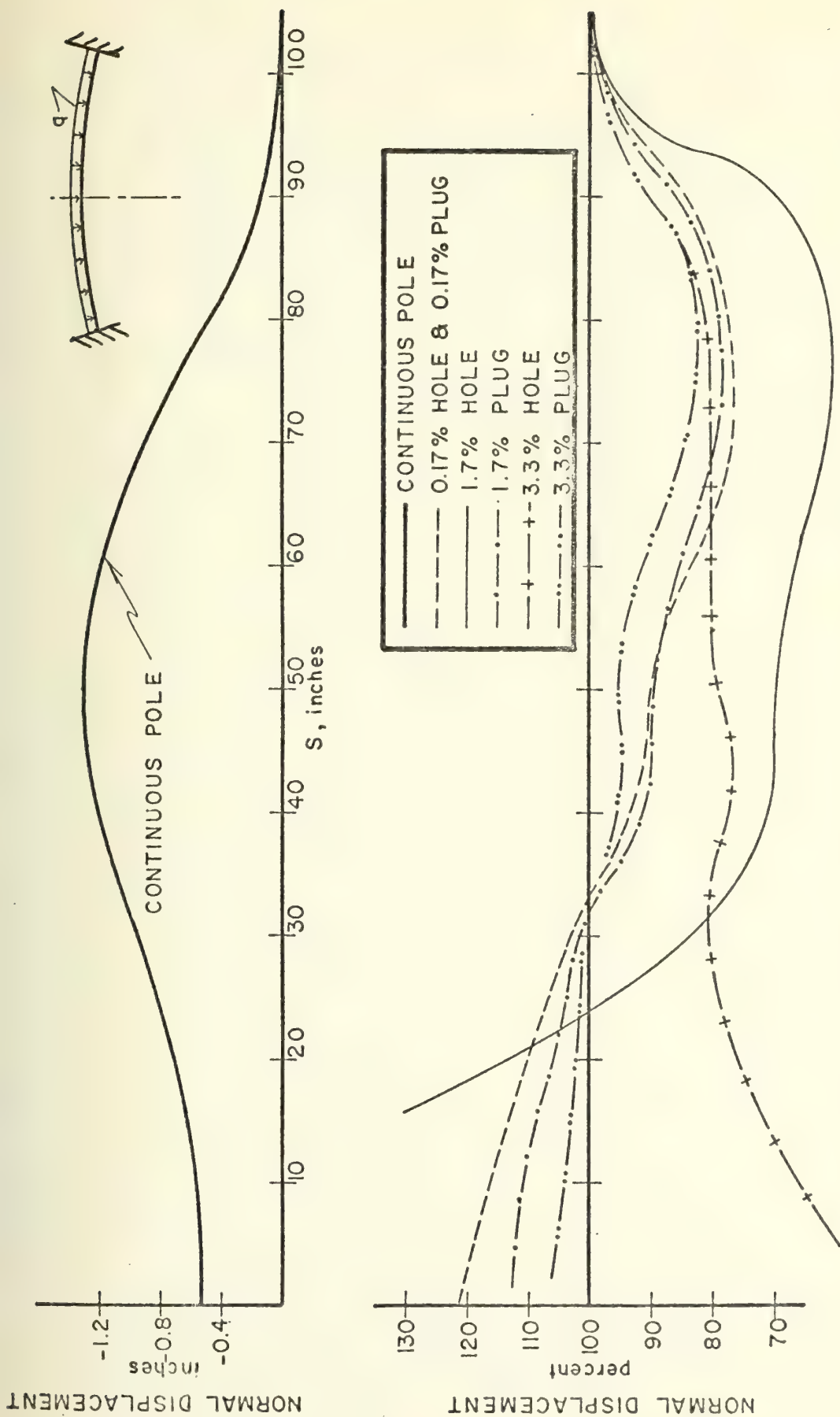
TABLE XI

SHALLOW CAP ($\lambda=6$) BUCKLING LOADS AND DISPLACEMENTS AT A POINT

ANALYSIS TYPE	POLE CONDITION	MERIDIONAL STATIONS/HARMONICS	q/q_0	NORMAL DEFLECTION AT 47.5% MERIDIAN (inches)
Axisymmetric Buckling - Symmetric Load	Pole	-	0.9950*	-
		30/1	1.0610	1.3720
		100/1	1.0200	1.3200
	Hole	-0.17%	100/1	1.1600
		-1.70%	100/1	1.0560
		-3.30%	100/1	1.0290
	Plug	-0.17%	100/1	1.2570
		-1.70%	100/1	1.1630
		-3.30%	100/1	1.2290
Bifurcation Buckling - Symmetric Load	Pole	-	0.7750*	-
		30/3	0.7851	0.6386
		66/3	0.7752	0.6188
	Hole	-0.17%	66/3	0.6356
		-1.70%	66/3	0.6562
		-3.30%	66/3	0.6264
	Plug	-0.17%	66/3	0.6356
		-1.70%	66/3	0.6438
		-3.30%	66/3	0.6513
Collapse Analysis - Asymmetric Load	Pole	-	0.7100†	-
		40/5	0.5391	0.9830
	Hole	-0.17%	40/5	1.0780
		-1.70%	40/5	1.0060
		-3.30%	40/5	1.4110
	Plug	-0.17%	40/5	0.9478
		-1.70%	40/5	0.8100
		-3.30%	40/5	0.7248

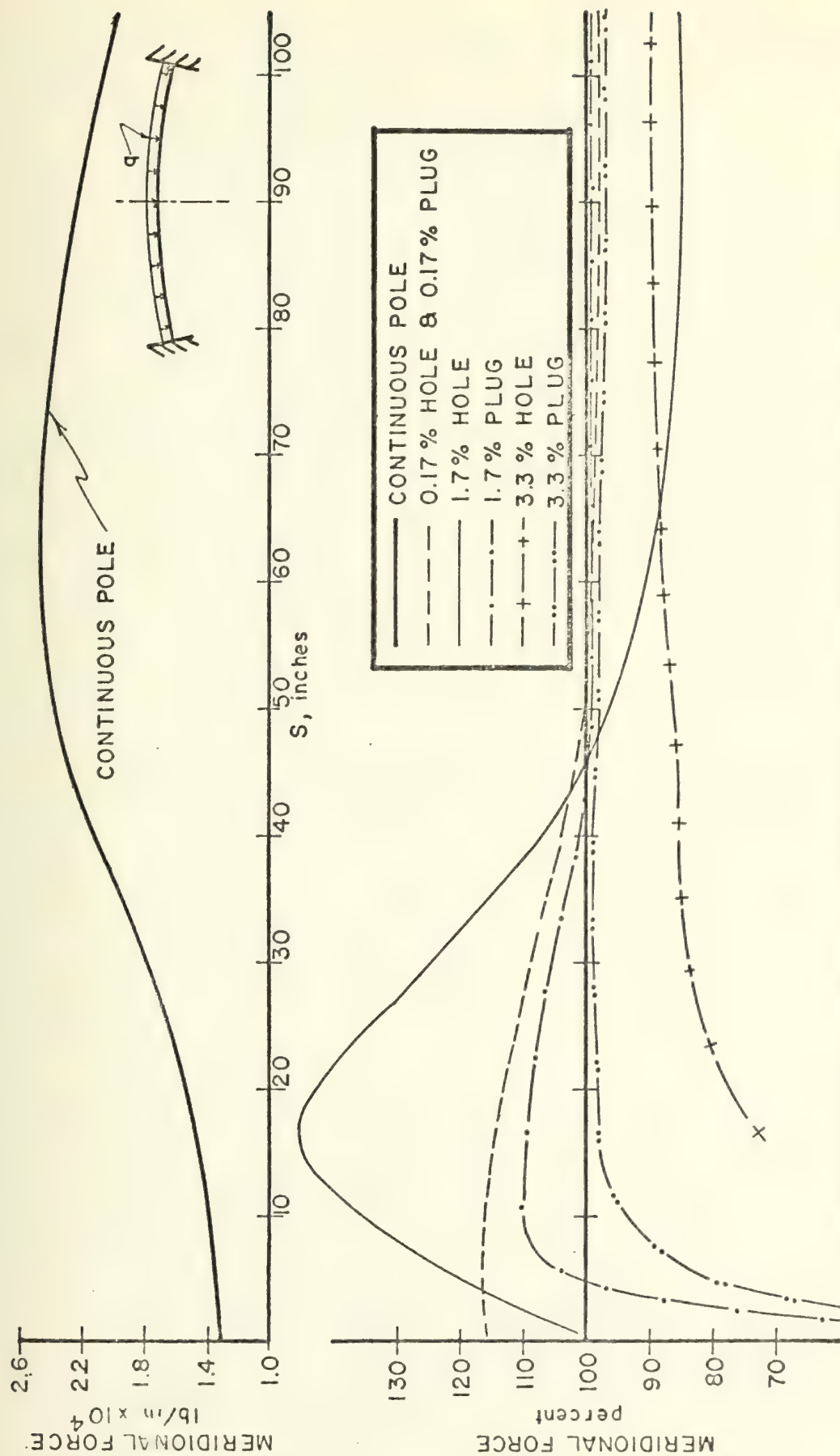
* Huang /Ref. 16/

† Famili and Archer /Ref. 17/



NORMAL DISPLACEMENT ERROR, AXISYMMETRIC ANALYSIS, $\lambda=6$ CAP

FIGURE 21



MERIDIONAL FORCE ERROR, AXISYMMETRIC ANALYSIS, $\lambda=6$ CAP

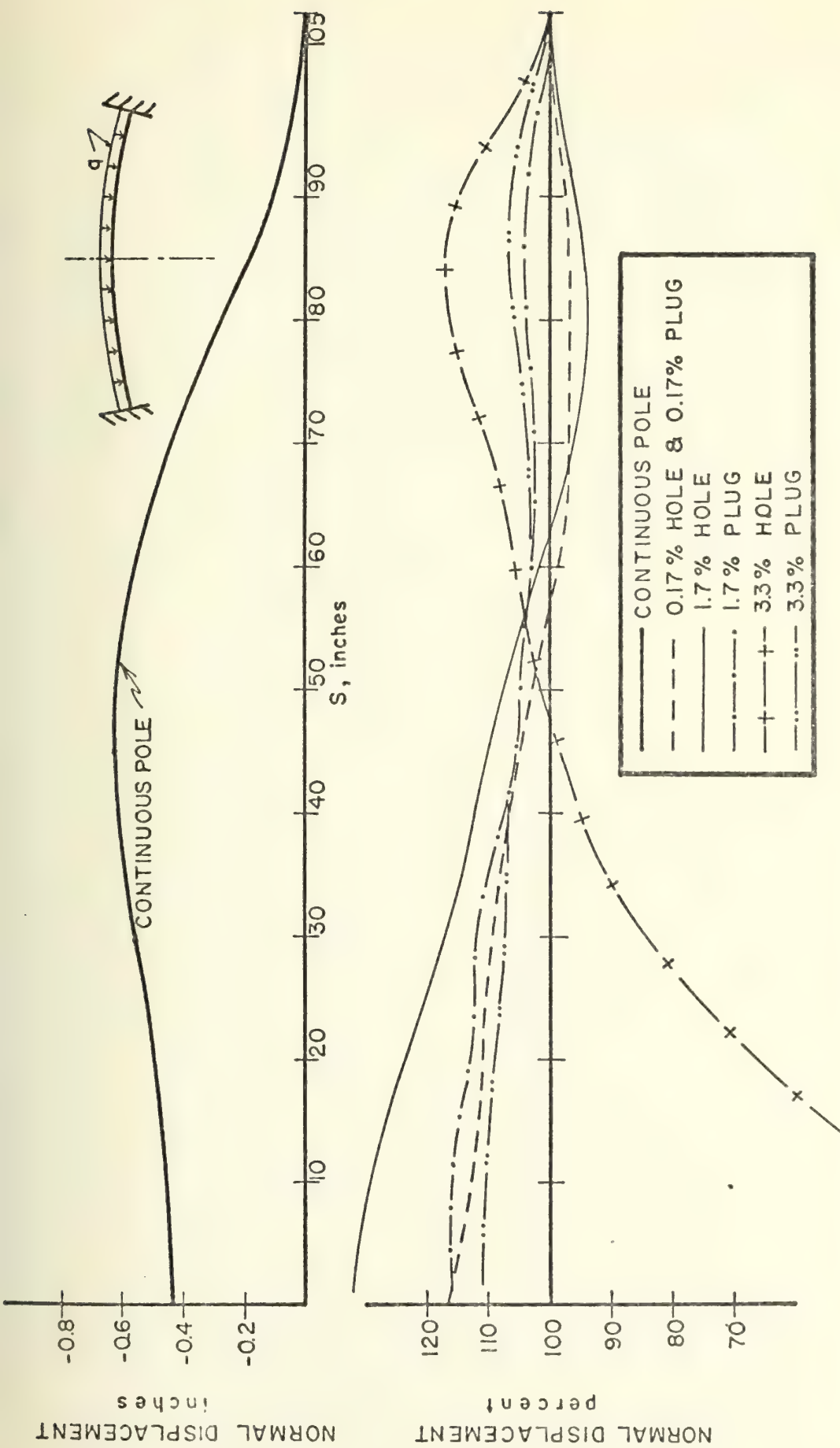
FIGURE 22

the meridian and were an average of 10% or less for the remainder of the meridian.

2. Bifurcation Buckling Analysis

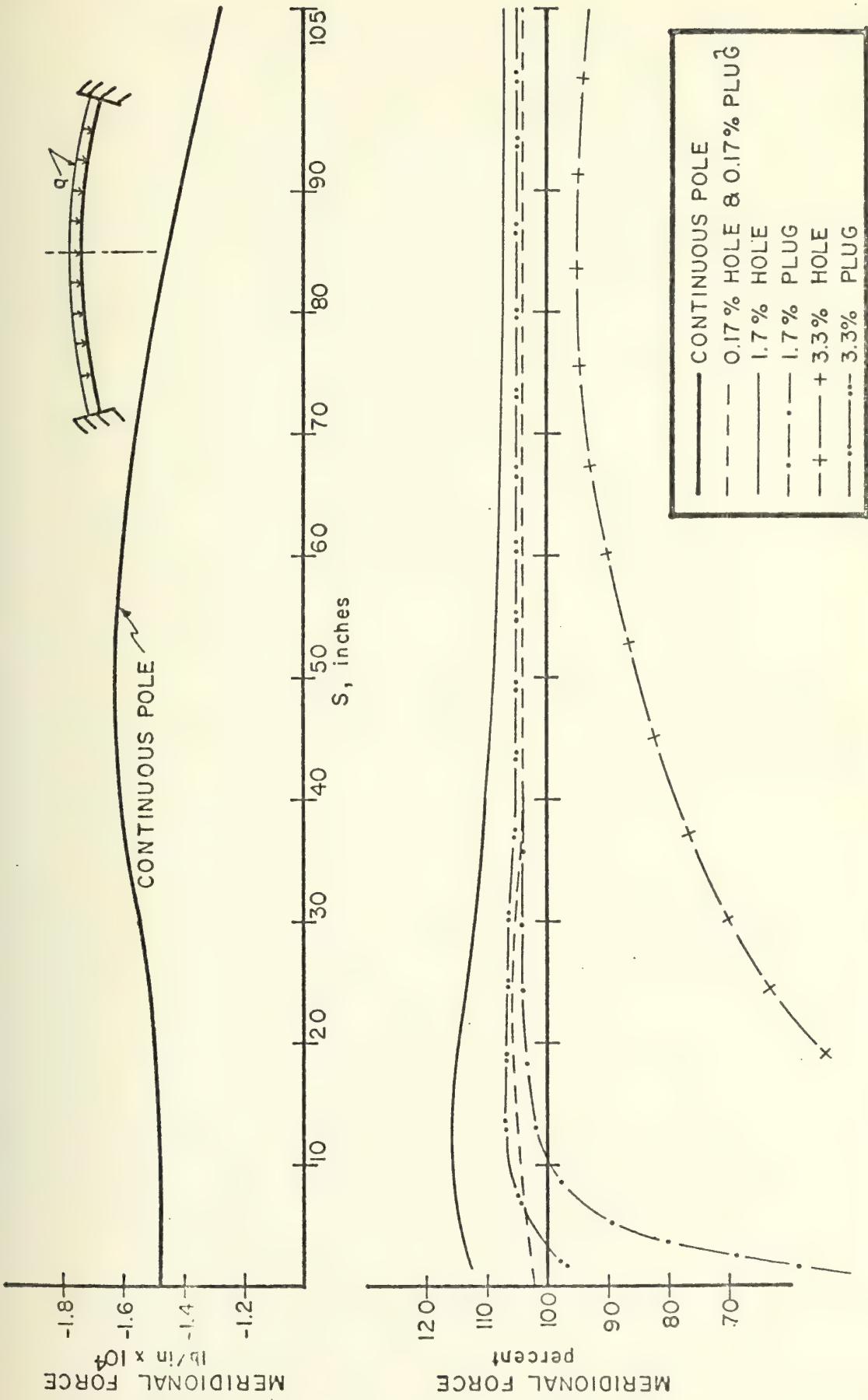
In this analysis, the shell buckling load appears to be insensitive to artificial pole conditions for the case of the plugs, and shows some sensitivity in the case of the holes. This is surprising since bifurcation buckling of the $\lambda = 6$ cap occurs in the $\text{Cos}(2\theta)$ mode. While equations (AD-8) show that no rotations exist at the pole in this mode, rotations certainly do exist at 1.7% and 3.3% meridian due to the buckling mode. The equations for the rigid plug's boundary conditions (equations (AD-10)) specifically prevent this rotation, while those for the hole (equations (AD-9)) do not.

Examination of Figures 23 and 24 show that the displacement and force distributions of the plugs at incipient buckling to be less erroneous, again, than the poles. In Figure 23, an average 10% displacement error occurs over the entire shell for the plugs, while holes of 1.7% and 3.3% meridian cause large errors over much of the shell. This trend is repeated in Figure 24, although the errors are not as large. From this analysis, the indications are that the 1.7% hole is a poor artificial pole and a 3.3% hole is unacceptable as it causes a 4.6% error in buckling load and over 40% error in some regions in both normal displacement and meridional force distributions. Larger hole sizes would apparently increase these errors.



NORMAL DISPLACEMENT ERROR, BIFURCATION BUCKLING ANALYSIS, $\lambda = 6$ CAP

FIGURE 23



MERIDIONAL FORCE ERROR, BIFURCATION BUCKLING ANALYSIS, $\lambda = 6$ CAP

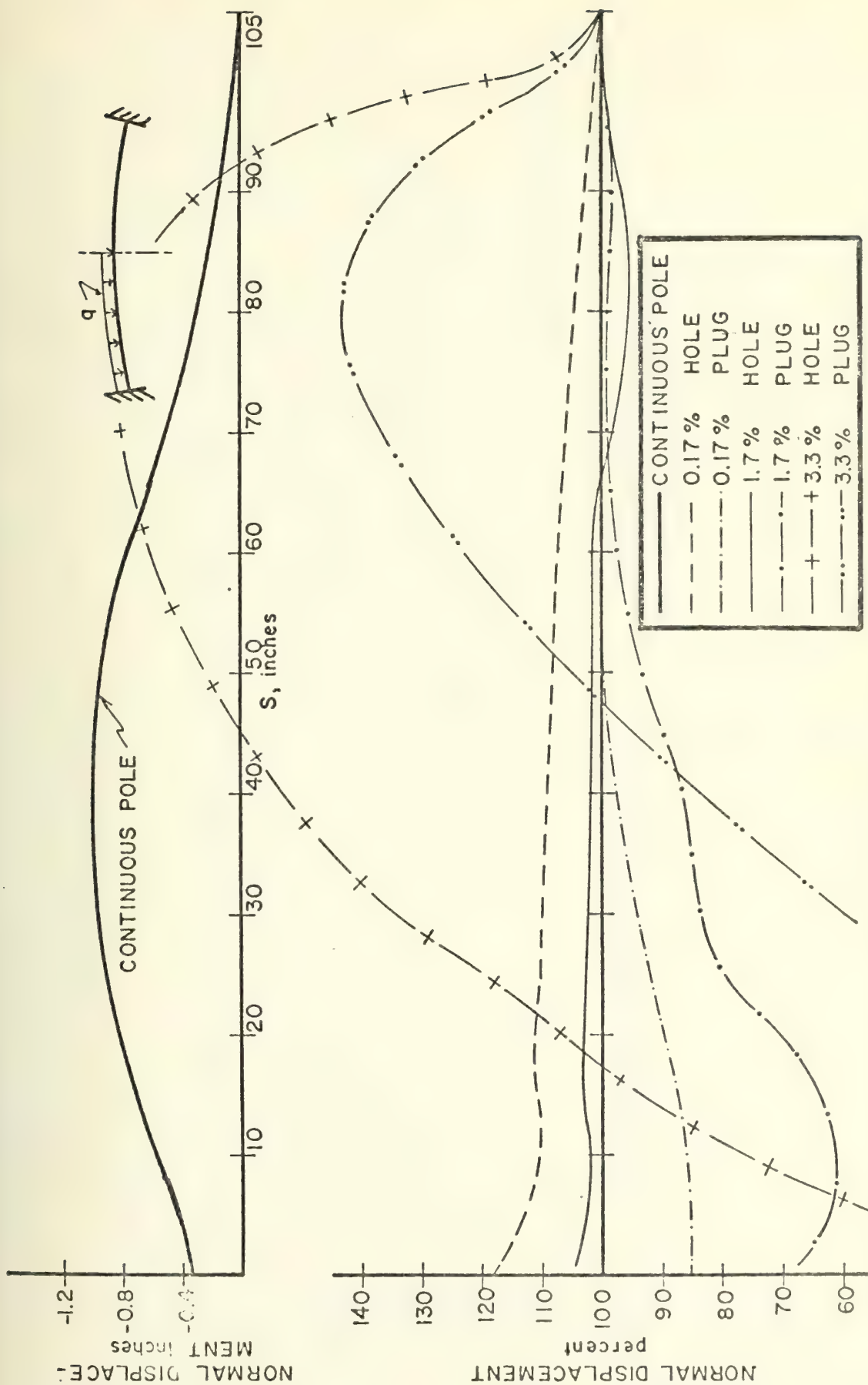
FIGURE 24

3. Nonsymmetric Collapse Analysis

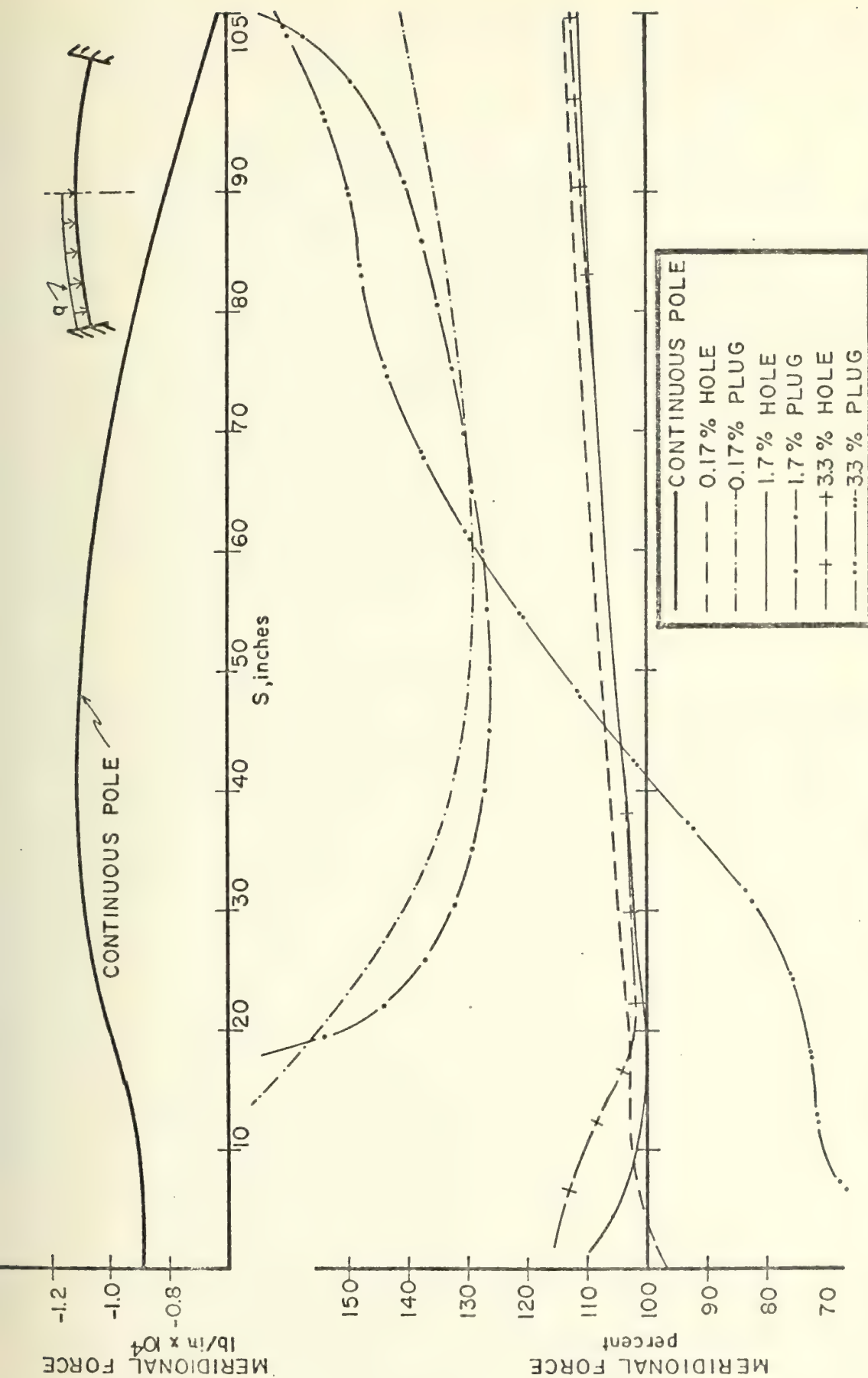
The results for the collapse analysis are also presented in Table XI, and in Figures 25 and 26 for the $\theta = 0^\circ$ meridian, which is symmetrically located under the loaded region. In Table XI, it is shown that the buckling loads of the cap with rigid plugs are quite erroneous (10.8%, 16.6% and 41.4% for the 0.17%, 1.7% and 3.3% plugs, respectively) while those for the small holes are quite reasonable. In Figures 25 and 26, it is seen that the small holes cause an average error of 10% in both the normal displacement and meridional force distributions at incipient buckling over the entire shell, while the errors caused by the rigid plugs is large.

In an asymmetrically loaded shell, large rotations will exist at the pole and the restrictions imposed by the rigid plug appear to make it an unacceptable artificial pole for use in this case.

Figures 25 and 26 also indicate that a 1.7% hole is the best artificial pole for this case. Assuming such a 'best' hole size exists for another given shell or loading condition, an investigation similar to that presented here would have to be performed to find it. This is an inherently expensive and time consuming process and there is no guarantee that there is a 'best' hole size without its radius becoming so small that mathematical difficulties arise during the solution process. Consequently, the use of



NORMAL DISPLACEMENT ERROR, NONSYMMETRIC COLLAPSE ANALYSIS, $\lambda=6$ CAP
FIGURE 25



MERIDIONAL FORCE ERROR, NONSYMMETRIC COLLAPSE ANALYSIS, $\lambda=6$ CAP

FIGURE 26

artificial pole conditions in nonsymmetric problems must be approached with caution.

B. THE $\lambda = 18$ SHALLOW CAP

A deeper shell with $\lambda = 18$ was investigated in a similar fashion and produced similar results. The shell buckling load was negligibly affected (except in those cases already mentioned for the $\lambda = 6$ shell) but displacement errors of up to 30% and meridional membrane force errors of an average 10-15% were noted. In general, the $\lambda = 18$ shell appeared less sensitive to the type of artificial pole (except in the asymmetric case), but as sensitive to any artificial pole as the shallower $\lambda = 6$ shell.

C. COMPARISONS WITH OTHER RESEARCHERS

As indicated in the introduction, various SATANS results were checked for validity with BOSOR and STAGS. These comparisons are presented with others in Table XII and very favorable agreement is seen to exist.

TABLE XII

SHALLOW CAP ($\lambda=6$) BUCKLING LOADS COMPARISON

ANALYSIS TYPE	POLE CONDITION	SOURCE	q/q_0	REMARKS
Axisymmetric Buckling	Pole	Huang SATANS	0.9950	Classical /Ref. 167
			1.0200	100 Stations
Bifurcation Buckling	Pole	Huang BOSOR SATANS	0.7750	50 Stations /Ref. 167
			0.7756	40 Stations /Ref. 157
			0.7752	66 Stations, 3 harmonics
			0.7689	30x36 variably spaced grid /Ref. 157
	0.17% Hole	STAGS BOSOR SATANS	0.7736	40 Stations
			0.7719	66 Stations, 3 harmonics
			0.7711	30x36 variable spaced grid
			0.7747	40 Stations
	1.70% Plug	STAGS BOSOR SATANS	0.7716	66 Stations, 3 harmonics
			0.7100	7 Stations /Ref. 177
			0.6730	7 Stations, 4 harmonics /Ref. 187*
			0.6560	7 Stations, 10 harmonics *
Nonsymmetric Collapse	Pole	Famili & Archer SATANS SATANS SATANS SATANS	0.5380	20 Stations, 4 harmonics *
			0.5380	20 Stations, 10 harmonics *
			0.5391	40 Stations, 5 harmonics
			0.5233	22x92 variable spaced grid
	1.70% Hole	STAGS SATANS	0.5389	40 Stations, 5 harmonics

* This result and the following three were taken from a discussion of the effects of the number of stations and harmonics on the buckling loads of the clamped spherical cap, contained in Ref. 18.

IV. CONCLUSIONS

The stability of the symmetrically loaded clamped shallow spherical shell appears to be fairly insensitive to artificial pole conditions in the axisymmetric and bifurcation buckling cases if the radius of the artificiality is kept at or below 3% of the meridian. In the case of the asymmetrically loaded shell, this insensitivity only applies when small holes are substituted for the pole. In general, the following additional conclusions can be drawn.

1. Sizable errors are introduced into the normal displacement and meridional membrane force distributions by the introduction of an artificial pole.
2. The displacement and force error distributions are sensitive to the size and type of artificial pole and are usually greater than 10% with some cases of over 30%, and they exist over the entire shell meridian.
3. Small rigid plugs appear to be the best polar models for symmetrically loaded shells. Apparently their continuity-preserving characteristics greatly outweigh their rotational-restricting characteristics.
4. Small holes appear to be the best polar models for asymmetrically loaded shells. Here, the rotational restrictions of the rigid plug make its use unacceptable. For this type of loading, an investigation will be required to determine the optimum hole size.

5. In general, the greater the displacement, or the higher the membrane force that would exist at a pole, the greater the error introduced by the use of an artificial pole.

APPENDIX A
NONLINEAR TERMS

The nonlinear terms of equations (15) are given below. Terms involving the initial imperfection rotation terms along have been omitted as they cause no straining.

$$\begin{aligned}
 \phi_S^2 &= (\sigma_0/E_0)^2 \left\{ \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right)^2 + \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right)^2 \right. \\
 &\quad + 2 \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right) \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right) \\
 &\quad + 2 \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right) \left(\sum_0^\infty \bar{\phi}_S^1(n) \cos n\theta \right) \\
 &\quad + 2 \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right) \left(\sum_1^\infty \bar{\phi}_S^2(n) \sin n\theta \right) \\
 &\quad + 2 \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right) \left(\sum_0^\infty \bar{\phi}_S^1(n) \cos n\theta \right) \\
 &\quad \left. + 2 \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right) \left(\sum_1^\infty \bar{\phi}_S^2(n) \sin n\theta \right) \right\} \\
 &\equiv \sigma_0/E_0 \left(\sum_0^\infty \beta_S^1(n) \cos n\theta + \sum_1^\infty \beta_S^2(n) \sin n\theta \right)
 \end{aligned} \tag{A-1}$$

$$\begin{aligned}
\Phi_{\theta}^2 &= (\sigma_0/E_0)^2 \left\{ \left(\sum_1^{\infty} \phi_{\theta}^1(n) \sin n\theta \right)^2 + \left(\sum_0^{\infty} \phi_{\theta}^2(n) \cos n\theta \right)^2 \right. \\
&\quad + 2 \left(\sum_1^{\infty} \phi_{\theta}^1(n) \sin n\theta \right) \left(\sum_0^{\infty} \phi_{\theta}^2(n) \cos n\theta \right) \\
&\quad + 2 \left(\sum_1^{\infty} \phi_{\theta}^1(n) \sin n\theta \right) \left(\sum_1^{\infty} \bar{\phi}_{\theta}^1(n) \sin n\theta \right) \\
&\quad + 2 \left(\sum_1^{\infty} \phi_{\theta}^1(n) \sin n\theta \right) \left(\sum_0^{\infty} \bar{\phi}_{\theta}^2(n) \cos n\theta \right) \\
&\quad + 2 \left(\sum_0^{\infty} \phi_{\theta}^2(n) \cos n\theta \right) \left(\sum_1^{\infty} \bar{\phi}_{\theta}^1(n) \sin n\theta \right) \\
&\quad \left. + 2 \left(\sum_0^{\infty} \phi_{\theta}^2(n) \cos n\theta \right) \left(\sum_0^{\infty} \bar{\phi}_{\theta}^2(n) \cos n\theta \right) \right\} \\
&\equiv \sigma_0/E_0 \left(\sum_0^{\infty} \beta_{\theta}^1(n) \cos n\theta + \sum_1^{\infty} \beta_{\theta}^2(n) \sin n\theta \right)
\end{aligned} \tag{A-2}$$

$$\begin{aligned}
\Phi^2 &= (\sigma_0/E_0)^2 \left\{ \left(\sum_1^{\infty} \phi^1(n) \sin n\theta \right)^2 + \left(\sum_0^{\infty} \phi^2(n) \cos n\theta \right)^2 \right. \\
&\quad \left. + 2 \left(\sum_1^{\infty} \phi^1(n) \sin n\theta \right) \left(\sum_0^{\infty} \phi^2(n) \cos n\theta \right) \right\} \\
&\equiv \sigma_0/E_0 \left(\sum_1^{\infty} \beta^1(n) \sin n\theta + \sum_0^{\infty} \beta^2(n) \cos n\theta \right)
\end{aligned} \tag{A-3}$$

$$\begin{aligned}
\Phi_S \Phi_\theta &= (\sigma_0/E_0)^2 \left(\left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right) \left(\sum_1^\infty \phi_\theta^1(n) \sin n\theta \right) \right. \\
&+ \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right) \left(\sum_0^\infty \phi_\theta^2(n) \cos n\theta \right) \\
&+ \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right) \left(\sum_1^\infty \phi_\theta^1(n) \sin n\theta \right) \\
&+ \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right) \left(\sum_0^\infty \phi_\theta^2(n) \cos n\theta \right) \\
&+ \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right) \left(\sum_1^\infty \bar{\phi}_\theta^1(n) \sin n\theta \right) \\
&+ \left(\sum_0^\infty \phi_S^1(n) \cos n\theta \right) \left(\sum_0^\infty \bar{\phi}_\theta^2(n) \cos n\theta \right) \\
&+ \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right) \left(\sum_1^\infty \bar{\phi}_\theta^1(n) \sin n\theta \right) \quad (A-4) \\
&+ \left(\sum_1^\infty \phi_S^2(n) \sin n\theta \right) \left(\sum_0^\infty \bar{\phi}_\theta^2(n) \cos n\theta \right) \\
&+ \left(\sum_0^\infty \bar{\phi}_S^1(n) \cos n\theta \right) \left(\sum_1^\infty \phi_\theta^1(n) \sin n\theta \right) \\
&+ \left(\sum_0^\infty \bar{\phi}_S^1(n) \cos n\theta \right) \left(\sum_0^\infty \phi_\theta^2(n) \cos n\theta \right) \\
&+ \left(\sum_1^\infty \bar{\phi}_S^2(n) \sin n\theta \right) \left(\sum_1^\infty \phi_\theta^1(n) \sin n\theta \right) \\
&+ \left. \left(\sum_1^\infty \bar{\phi}_S^2(n) \sin n\theta \right) \left(\sum_0^\infty \phi_\theta^2(n) \cos n\theta \right) \right) \\
&\equiv \sigma_0/E_0 \left(\sum_0^\infty \beta_{S\theta}^1(n) \cos n\theta + \sum_1^\infty \beta_{S\theta}^2(n) \sin n\theta \right)
\end{aligned}$$

APPENDIX B

EIGENVALUE PROBLEM MATRIX ELEMENT DEFINITIONS

The elements of the $\langle \bar{A} \rangle$ and $\langle \bar{B} \rangle$ matrices used in the eigenvalue solution are defined below.

$$\begin{aligned}
 A_{11} &= G_{11} - E_{11}Q^2 \\
 A_{12} &= F_{12}Q \\
 A_{13} &= F_{13}Q \\
 A_{21} &= -F_{21}Q \\
 A_{22} &= -E_{22}Q^2 + G_{22} - G_{24}G_{42}/G_{44} \\
 A_{23} &= G_{23} + E_{23}Q^2 - G_{24}(G_{43} - E_{43}Q^2)/G_{44} \\
 A_{31} &= -F_{31}Q \\
 A_{32} &= G_{32} - E_{32}Q^2 - G_{42}(G_{34} - E_{34}Q^2)/G_{44} \\
 A_{33} &= G_{33} - E_{33}Q^2 - (G_{34} - E_{34}Q^2)(G_{43} - E_{43}Q^2)/G_{44}
 \end{aligned} \tag{B-1}$$

where

$$Q = m\pi/L$$

and where

$$\begin{aligned}
 B_{11} &= n^2/4 \\
 B_{12} &= nQ/4 \\
 B_{13} &= 0 \\
 B_{21} &= n^2/4 \\
 B_{22} &= Q^2/4 \\
 B_{23} &= 0 \\
 B_{31} &= 0 \\
 B_{32} &= 0 \\
 B_{33} &= Q^2
 \end{aligned} \tag{B-2}$$

and where all elements of the 4 x 4 matrices \underline{E} , \underline{F} and \underline{G} are zero except for the following, defined as

$$\begin{aligned}
 E_{11} &= b \\
 E_{22} &= \frac{1}{2}(1-\nu)b + 9(1-\nu)d/(8R_0^2) \\
 E_{23} &= 3(1-\nu)dn/(2R_0^2) \\
 E_{32} &= E_{23} \\
 E_{33} &= 2(1-\nu)dn^2/R_0^2 \\
 E_{34} &= 1 \\
 E_{43} &= -d
 \end{aligned}
 \tag{B-3}$$

$$\begin{aligned}
 F_{12} &= \frac{1}{2}(1-\nu)bn/R_0 - 3(1-\nu)dn/(8R_0^3) \\
 F_{13} &= bv/R_0 - \frac{1}{2}(1-\nu)n^2/R_0^3 \\
 F_{21} &= -F_{12} \\
 F_{31} &= -F_{13}
 \end{aligned}
 \tag{B-4}$$

$$\begin{aligned}
 G_{11} &= -\frac{1}{2}(1-\nu)bn^2/R_0^2 - (1-\nu)dn^2/(8R_0^4) \\
 G_{22} &= -bn^2/R_0^2 - (1-\nu)(1+\nu)dn^2/R_0^4 \\
 G_{23} &= -bn/R_0^2 - (1-\nu)(1+\nu)dn^3/R_0^4 \\
 G_{24} &= -vn/R_0^2 \\
 G_{32} &= G_{23} \\
 G_{33} &= -vn^2/R_0^2 \\
 G_{42} &= vdn/R_0^2 \\
 G_{43} &= vdn^2/R_0^2 \\
 G_{44} &= -1
 \end{aligned}
 \tag{B-5}$$

and where

$$\begin{aligned}
 b &= \{(1-\nu)(1+\nu)\}^{-1} \\
 d &= (b/12)(h/r)^2
 \end{aligned}
 \tag{B-6}$$

APPENDIX C

INPUT DATA GUIDE FOR SATANS-I AND SATANS-II

<u>Card</u>	<u>Columns</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
1	1-72	18A4	TITLE	-	Enter any 72 characters

2	1- 5	I5	NO	1	The problem number, 0<NO<10000.
2	6-10	L5	\$DYNAMC	F T	For a static analysis, set \$DYNAMC = F. For a dynamic analysis, set \$DYNAMC = T.
2	11-15	I5	IMODE	0 1	For no modal output data. For modal output data.
2	16-20	I5	NDIMEN	0 1	Dimensional output data. Nondimensional output.
2	21-25	I5	NTHMAX	8	Summed solution will be printed at NTHMAX merid- ians, 0<=NTHMAX<=36.
2	26-30	I5	IFREQ	3	Solution will be printed at the first station, every subsequent IFREQ station and the last sta- tion, 0<IFREQ<=KMAX.
2	31-35	I5	IPRINT	2	Every IPRINT converged solution will be print- ed.
2	36-40	I5	IBCINL	-1 0	If the shell has a pole at the first station. If the shell has no pole at the first station.
2	41-45	I5	IBCFNL	-1 0	If the shell has a pole at the last station. If the shell has no pole at the last station.

<u>Card</u>	<u>Columns</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
2	46-50	I5	KMAX	35	Number of meridional stations. Note: KMAX<201 for SATANS-I without plots and KMAX<101 for SATANS-I with plots or for SATANS-II.
2	51-55	I5	MNMAX	7	Number of series coefficients used to describe the initial conditions, pressure and thermal loads (and initial imperfections if using SATANS-II). Note: For SATANS-I: 0<MAXM<11 and KMAX*MAXM<201. For SATANS-II: 0<MAXM<26 and KMAX*MAXM<951.
2	61-65	I5	LSMAX	1 99 750	For a linear analysis. Use many load steps for a nonlinear static analysis. For a dynamic analysis, LSMAX is the number of time increments, where $LSMAX = T_{max} / \Delta T$.
2	66-70	I5	LCHMAX	3 0	The number of load step size reductions in a static analysis, recommended range = 2-4. For a dynamic analysis.
2	71-75	I5	ITRMAX	1 30	For a linear analysis. The number of iterations at a load or time step. For a nonlinear analysis, suggested range = 10-30, up to 50 for special cases.
2	76-80	I5	IC	0 1	Initial conditions. Set to 0 for a static analysis, or for a dynamic analysis where the shell is at rest at $T=0$. For a dynamic analysis with initial conditions.

<u>Card</u>	<u>Columns</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
3	1-12	E12.3	NU	0.3	Poisson's ratio, ν .
3	12-24	E12.3	SIGO	1000.0 1.0	Reference stress level, σ_0 . If the input data is dimensional.
3	24-36	E12.3	ELAST	.3E8 1.0	Reference modulus of elasticity, E_0 . If the input data is dimensional.
3	37-48	E12.3	TKN	.4E-2 1.0	Reference thickness, h_0 . If the input data is dimensional.
3	49-60	E12.3	CHAR	8.16 1.0	Characteristic shell dimension, a . If the input data is dimensional.
3	61-72	E12.3	TEEO	0.0 .996E-5	If a static analysis. Reference time, T_0 .
<hr/>					
4	1-12	E12.3	DELOAD	0.2 .1823E-6	For a static analysis, DELOAD is the load increment. It remains unchanged until the solution fails to converge in ITRMAX iterations, when it is reduced by a factor of five. A maximum of LCHMAX such reductions will occur. For a dynamic analysis, DELOAD is the nondimensional time increment.
4	13-24	E12.3	EPS	0.01	The convergence criterion recommended range of $0.01 < EPS < 0.001$.

Card 4A is only included for a SATANS-II analysis.

4A	1- 5	I5	JUMP	1	For an analysis using single series expansions.
				2	For an analysis using double series expansions.

<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
4A	5-10	I5	MPERFS	0	An analysis without im- perfections.
				1	An analysis with imperfec- tions. Note: if JUMP=2, MPERFS may be 0 or 1. if JUMP=1, MPERFS must be 0 if MPERFS=1, JUMP must be 2.

Include as many cards 5 as necessary to specify NTHMAX
meridians. If NTHMAX equals 0, omit card 5.

5	1-72	6E12.3	10.0	A list of circumferential coordinates θ , in degrees and tenths, where the solution printout is de- sired. The list must have NTHMAX entries.
---	------	--------	------	---

If IBCINL=-1, omit cards 6 through 14. If IBCFNL=-1, omit
cards 15 through 23. Cards 6 through 23 describe the
boundary conditions at the first, and then at the last sta-
tion. The boundary conditions exist on the total variables,
not on the individual harmonics. Loadings applied through
specification of boundary conditions are taken in the zeroeth
harmonic (n=0) only, as the column matrix { Ω } is set to zero
for harmonics greater than zero. The boundary conditions are
dimensional. The format of cards 6 through 23 is 4E16.8.

<u>Card 6,15</u>	<u>Card 7,16</u>	<u>Card 8,17</u>	<u>Card 9, 18</u>	
$\Omega(1,1)$	$\Omega(1,2)$	$\Omega(1,3)$	$\Omega(1,4)$	$\begin{pmatrix} N_S \\ N_{S\theta} \\ Q_S \\ \Phi_S \end{pmatrix} +$
$\Omega(2,1)$	$\Omega(2,2)$	$\Omega(2,3)$	$\Omega(2,4)$	
$\Omega(3,1)$	$\Omega(3,2)$	$\Omega(3,3)$	$\Omega(3,4)$	
$\Omega(4,1)$	$\Omega(4,2)$	$\Omega(4,3)$	$\Omega(4,4)$	

Card 10,19	Card 11,20	Card 12,21	Card 13,22		Card 14,23
$\Lambda(1,1)$	$\Lambda(1,2)$	$\Lambda(1,3)$	$\Lambda(1,4)$	$\left[\begin{array}{c} U \\ V \\ W \\ M_s \end{array} \right]$	$= \left[\begin{array}{c} \ell(1) \\ \ell(2) \\ \ell(3) \\ \ell(4) \end{array} \right]$
$\Lambda(2,1)$	$\Lambda(2,2)$	$\Lambda(2,3)$	$\Lambda(2,4)$		
$\Lambda(3,1)$	$\Lambda(3,2)$	$\Lambda(3,3)$	$\Lambda(3,4)$		
$\Lambda(4,1)$	$\Lambda(4,2)$	$\Lambda(4,3)$	$\Lambda(4,4)$		

Card 24 is:

1. Included for a SATANS-I static analysis.
2. Included but blank for a SATANS-I dynamic analysis.
3. Omitted for a SATANS-II analysis.

Card	Column	Format	Item	Example	Meaning
24	1- 2	L2	\$PLOTS	F	Indicates plots are NOT desired.
				T	Indicates plots are desired.
24	3- 4	L2	\$MODAL	F	Indicates plots are for summed solutions only
				T	Indicates plots are for modal solutions only.

For the remainder of card 24 entries, 0 indicates that no plots are desired for the particular item, and 1 indicates that they are desired. All graphs are plotted as the indicated item versus the station number. If a complete plot is desired, insure IFREQ=1.

24	5- 6	I2	IRADII	1	Plot the radii as computed by subroutine GEOM.
24	7- 8	I2	IGAMMA	1	Plot ρ'/ρ as computed by subroutine GEOM.
24	9-10	I2	IOMEGS	1	Plot ω_s as computed by subroutine GEOM.
24	11-12	I2	IOMEGT	1	Plot ω_θ as computed by subroutine GEOM.
24	13-14	I2	IDEOMS	1	Plot ω'_s as computed by subroutine GEOM.
24	15-16	I2	IBSTIF	1	Plot the stiffness d as computed by subroutine BDB.

<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
24	17-18	I2	IDSTIF	1	Plot the stiffness d as computed by the subroutine BDB.
24	19-20	I2	IBBSTF	1	Plot the stiffness $db/d\zeta$ as computed by subroutine BDB.
24	21-22	I2	IDDSTF	1	Plot the stiffness $dd/d\zeta$ as computed by subroutine BDB.
24	23-24	I2	IPR	1	Plot the normal component of the pressure load.
24	25-26	I2	IPS	1	Plot the meridional component of the pressure load.
24	27-28	I2	IPT	1	Plot the circumferential component of the pressure load.
24	29-30	I2	ITT	1	Plot the thermal load.
24	31-32	I2	IMT	1	Plot the thermal moment.
24	33-34	I2	IDTT	1	Plot $d/d\zeta$ of the thermal load.
24	35-36	I2	IDMT	1	Plot $d/d\zeta$ of the thermal moment.
24	37-38	I2	INS	1	Plot the meridional membrane force distribution.
24	39-40	I2	INTH	1	Plot the circumferential membrane force distribution
24	41-42	I2	INSTH	1	Plot the meridio-circumferential membrane force distribution.
24	43-44	I2	IQS	1	Plot the transverse force distribution.
24	45-46	I2	IMS	1	Plot the meridional moment distribution.
24	47-48	I2	IMTH	1	Plot the circumferential moment distribution.

<u>Card</u>	<u>Column</u>	<u>Format</u>	<u>Item</u>	<u>Example</u>	<u>Meaning</u>
24	49-50	I2	IMSTH	1	Plot the meridio-circumferential moment distribution.
24	51-52	I2	IU	1	Plot the meridional displacement distribution.
24	53-54	I2	IV	1	Plot the circumferential displacement distribution.
24	55-56	I2	IW	1	Plot the normal displacement distribution.
24	57-58	I2	IPHIS	1	Plot the meridional rotation distribution.
24	59-60	I2	IPHIT	1	Plot the circumferential rotation distribution.
24	61-62	I2	IPHI	1	Plot the meridio-circumferential rotation distribution.

Insert imperfection data here for a SATANS-II analysis with imperfections. Insure format of the imperfection data is compatible with that specified in the user-written subroutine IMPERF.

25	1- 2	I2	IRNAGN	0	Indicates this is the only run.
				1	Indicates another run is to be made. Add another complete set of data cards after this card is IRNAGN=1.

APPENDIX D

LISTING OF SATANS-I USER PREPARED SUBROUTINES

```

C*****
C THIS PROGRAM SERVES AS THE 'MAIN' PROGRAM, AND CALLS SUBROUTINE
C 'SATANS' AT THE END OF A RUN, IT CHECKS 'IRNAGN' TO SEE IF
C THERE IS ANOTHER RUN.
C*****
COMMON /BLRUN/ IRNAGN
1 CALL SATANS
CALL FLYNVY
IF (IRNAGN.EQ.1) GO TO 1
STOP
END

```

```

SUBROUTINE GEJM
C*****
C THIS SUBROUTINE COMPUTES THE NONDIMENSIONAL GEOMETRY FUNCTIONS
C OF THE SHELL.
C*****
REAL NU, LAM, LAM2, JAY, MT, LSD18, LSD1N, MASS
COMMON /IBL4/ KMAX, KL
COMMON /BL8/ R(200), GAM(200), OMT(200)
COMMON /BL11/ DMXI(200), PHEE, TO, T2
COMMON /BL17/ DEL
COMMON /BL20/ DEOMX(200)
COMMON /BL32/ TKN, ELAST, CHAR, SIGO
COMMON /BL102/ DELOAD
COMMON /BL103/ MASS(200)
C*****
GEOMETRY FOR CYLINDER A-14
R(S) = INFINITY
R(TH) = 4.0 INCHES
S = 8.16 INCHES
KMAX = 35 STATIONS
A = 8.16 INCHES (CHARACTERISTIC SHELL DIMENSION)
C*****

```



```

DEL=8.16/(8.16*34.0)
RO=4.0/8.16
RORCP=1.0/RO
DO 1 K=1,KMAX
  R(K)=RO
  GAM(K)=0.0
  CMT(K)=RORCP
  OMXI(K)=0.0
  DEOMX(K)=0.0
1 RETURN
END

```

```

SUBROUTINE BD3(K,B,DB,D,DD)
C*****
C THIS SUBROUTINE COMPUTES THE NONDIMENSIONAL INPLANE AND BENDING
C STIFFNESSES OF THE SHELL.
C*****
REAL NU,LAM,LAM2,JAY,MT,LSD18,LSDIN,MASS
COMMON /BL15/ NU,U1(10),V1(10),W1(10),V2(10),W2(10),U3(10),
1 V3(10),W3(10)
COMMON /BL17/ DEL
COMMON /BL32/ TKN,ELAST,CHAR,SIG/
C*****
C NONDIMENSIONAL STIFFNESS QUANTITIES FOR CYLINDER A-14
C WHERE:
C E=15,800,000 PSI
C H(0)=15,800,000 PSI
C A=0.00437 INCHES
C B=1./12.
C DB=0.
C DD=0.
C RETURN
END

```



```

C*** SUBROUTINE PLDAD(K)
C*** THIS SUBROUTINE ESTABLISHES THE NONDIMENSIONAL FOURIER
C*** COEFFICIENTS OF THE LOADS APPLIED TO THE SHELL, AND ALSO
C*** ESTABLISHES THE MODES TO BE CONSIDERED IN THE ANALYSIS.
C*** ESTABLISHMENT OF MODES CAN ALSO BE ACCOMPLISHED IN SUBROUTINE
C*** TLOAD.
C*** COMMON /IBL1/ MNMAX
C*** COMMON /IBL2/ NN(10), MNINIT
C*** COMMON /IBL4/ KMAX, KL
C*** COMMON /INL8/ LSTEP, ITR
C*** COMMON /BL3/ PR(10), PX(10), PT(10)
C*** COMMON /BL6/ Z(4,220), SOE, OSE, ALOAD
C*** COMMON /BL8/ R(200), GAM(200), OMT(200)
C*** COMMON /BL32/ TKN, ELAST, CHAR, SIGO
C*** COMMON /BL102/ DELOAD
C*** COMMON /BL103/ MASS(200)
C*** THIS LOADING IS FOR ESTABLISHMENT OF THE CYLINDER A-14 BIFURCA-
C*** TION BUCKLING LOAD IN THE 21ST HARMONIC.
C*** THE PRESSURE LOADING DEFINED HERE ACTS AS MODAL TRIGGERS, AS THE
C*** AXIAL LOAD IS INTRODUCED THROUGH THE BOUNDARY CONDITIONS.
C*** A=CHAR/(SIGO*TKN)
C*** NN(1)=0
C*** NN(2)=21
C*** PR(1)=0.0000001#A
C*** PR(2)=0.0000001#A
C*** RETURN
C*** END

```


APPENDIX E

LISTING OF SATANS-I

THIS LISTING COMPRISES THE STORABLE PACKAGE OF SATANS-I AND CONTAINS THE FOLLOWING SUBROUTINES:

SATANS	STATIC	DYNAMC	OUTPUT
PLCT2	FLYNNV	FORCE	MATINV
ABC	PANDD	MODES	HJ
POLE	PLCTIT	SCALIT	DRAWIT
ROUND	XANDZ	PLOTI	PMATRX
UPDATE	INLPOL	FNLPOL	PHIBET
EFG	TEAETA		

```

SUBROUTINE SATANS
C*****
C THIS SUBROUTINE READS ALL DATA, PRINTS THE OUTPUT TITLE PAGE,
C AND PASSES CONTROL TO ONE OF THE MAJOR CONTROLLING SUBROUTINES
C*****
IMPLICIT LOGICAL#1 ($)
REAL#4 NU,LAM,LAM2,JAY,MT, LSD18,LSDIN,MASS
COMMON /IBL1/ MMAX
COMMON /IBL4/ KMAX,KL
COMMON /IBL5/ IBCINL,IBCFNL
COMMON /IBL9/ MAXM
COMMON /IBL10/ IFREQ,NTHMAX
COMMON /IBL13/ ITRMAX,LSMAX
COMMON /BL13/ OMEG1(4,4),CAPL1(4,4),CAPLL(4,4),
1 OMEG1(4,4)
1 COMMON /BL15/ UNIT(4,4)
COMMON /BL15/ VJ,U1(10),V1(10),W1(10),V2(10),U2(10),W2(10),U3(10),
1 V3(10),W3(10)
COMMON /BL16/ EPS
COMMON /BL18/ ELI(4),ELL(4)
COMMON /BL19/ TH(36)
COMMON /BL32/ TKN,ELAST,CHAR,SIGO
COMMON /BL100/ TTEO,$DYNMC
COMMON /BL102/ DELOAD
COMMON /BLPLOT/ ITRADII,IGAMMA,IOMEGS,IOMEGT,IDEOMS,IBSTIF,IDSTIF,
1 IBBSTF,IBDDSTF,IPT,IPT,IIT,IMT,IDTT,IMT,INS,
2 INTH,INSTH,IQS,IMS,IMTH,IMSTH,IU,IV,IW,IPHS,
3 IPHIT,IPHI,$PLOTS,$MODAL

```



```

COMMON /BLDATA/ TITLE,NO,IMODE,NDIMEN,IPRINT,LCHMAX,IC
COMMON /BLRUN/ IIRNAGN
DIMENSION TITLE (18)
C*****
C READ IN DATA FOR THIS RUN
C*****
READ (5,100) TITLE
READ (5,101) NO,$DYNMC,IMODE,NDIMEN,NTHMAX,IFREQ,IPRINT,IBCINL,
1 IBCFNL,KMAX,MNMAX,MAXM,LSMAX,LCHMAX,ITRMAX,IC
READ (5,102) NU,SIGO,ELAST,TKN,CHAR,TEEO
READ (5,103) DELOAD,EPS
IF (NTHMAX.EQ.0) GO TO 1
READ (5,104) (TH(NTH),NTH=1,NTHMAX)
DO 2 NTH=1,NTHMAX
2 TH(NTH)=TH(NTH)*6.2831853/360.0
1 IF (IBCINL.EQ.0) READ (5,105) OMEGL,CAPL1,ELL
IF (IBCFNL.EQ.0) READ (5,105) OMEGL,CAPL1,ELL
READ (5,106) $PLOTS,$MODAL,IRADII,IGAMMA,IOMEGS,IOMEGT,IDEOMS,
1 IBSTIF,IDSTIF,IBBSTIF,IBBSTIF,IPR,IPS,IPT,ITT,IMT,IDTT,
2 IDMT,INS,INTH,INSTH,IQS,IMS,IMTH,IMSTH,IU,IV,IW,
3 IPHIS,IPHIT,IPHI
READ (5,107) IIRNAGN
C*****
C THIS CONCLUDES DATA READ-IN FOR THIS RUN.
C PRINT OUT PERTINENT DATA.
C*****
WRITE (6,200) NO
WRITE (6,201) TITLE
WRITE (6,202)
IF (IBCINL.LT.0) WRITE (6,203)
IF (IBCFNL.LT.0) GO TO 3
WRITE (6,204)
WRITE (6,205) (OMEG1(1,J),J=1,4),(CAPL1(1,J),J=1,4),ELL(1)
WRITE (6,206) (OMEG1(2,J),J=1,4),(CAPL1(2,J),J=1,4),ELL(2)
WRITE (6,207) (OMEG1(3,J),J=1,4),(CAPL1(3,J),J=1,4),ELL(3)
WRITE (6,208) (OMEG1(4,J),J=1,4),(CAPL1(4,J),J=1,4),ELL(4)
3 IF (IBCFNL.LT.0) WRITE (6,209)
IF (IBCFNL.LT.0) GO TO 4
WRITE (6,210)
WRITE (6,211) (OMEG1(1,J),J=1,4),(CAPL1(1,J),J=1,4),ELL(1)
WRITE (6,212) (OMEG1(2,J),J=1,4),(CAPL1(2,J),J=1,4),ELL(2)
WRITE (6,213) (OMEG1(3,J),J=1,4),(CAPL1(3,J),J=1,4),ELL(3)
WRITE (6,214) (OMEG1(4,J),J=1,4),(CAPL1(4,J),J=1,4),ELL(4)
4 IF ($DYNMC) GO TO 5
WRITE (6,215) KMAX,MAXM,DELOAD,LSMAX,ITRMAX,LCHMAX,EPS
WRITE (6,216) CHAR,TKN,ELAST,SIGO,NU
IF (.NOT.$PLOTS) GO TO 6

```



```

1  ICHCK1=IABS(IRADII)+IABS(IGAMMA)+IABS(IOMEGS)+IABS(IOMEGT)
2  +IABS(IDEOMS)+IABS(IBSTIF)+IABS(IDSTIF)+IABS(IBBSTF)
3  +IABS(IDDSTF)+IABS(IPR)+IABS(IPS)+IABS(IPT)+IABS(ITT)
4  +IABS(IMT)+IABS(IDIT)+IABS(IDMT)
5  (ICHCK1.GT.0) WRITE (6,214)
6  IF (ICHCK1.EQ.0) GO TO 8
7  IF (ICHCK1.EQ.0) WRITE (6,215)
8  IF (IRADII.NE.0) WRITE (6,216)
9  IF (IGAMMA.NE.0) WRITE (6,217)
10 IF (IOMEGS.NE.0) WRITE (6,218)
11 IF (IOMEGT.NE.0) WRITE (6,219)
12 IF (IDEOMS.NE.0) WRITE (6,220)
13 IF (IBSTIF.NE.0) WRITE (6,221)
14 IF (IDSTIF.NE.0) WRITE (6,222)
15 IF (IBBSTF.NE.0) WRITE (6,223)
16 IF (IPR.NE.0) WRITE (6,224)
17 IF (IPS.NE.0) WRITE (6,225)
18 IF (IPT.NE.0) WRITE (6,226)
19 IF (ITT.NE.0) WRITE (6,227)
20 IF (IMT.NE.0) WRITE (6,228)
21 IF (IDIT.NE.0) WRITE (6,229)
22 IF (IDMT.NE.0) WRITE (6,230)
23 ICHCK1=IABS(INS)+IABS(INTH)+IABS(IQS)+IABS(IMS)
24 +IABS(IMTH)+IABS(IMSTH)+IABS(IU)+IABS(IV)+IABS(IW)
25 +IABS(IPHIS)+IABS(IPHIT)+IABS(IPHI)
26 IF (ICHCK1.GT.0) .AND. $MODAL) WRITE (6,231)
27 IF (ICHCK1.GT.0) .AND. .NOT. $MODAL) WRITE (6,232)
28 IF (ICHCK1.EQ.0) GO TO 9
29 IF (INS.NE.0) WRITE (6,233)
30 IF (INTH.NE.0) WRITE (6,234)
31 IF (IMSTH.NE.0) WRITE (6,235)
32 IF (IQS.NE.0) WRITE (6,236)
33 IF (IMTH.NE.0) WRITE (6,237)
34 IF (IMSTH.NE.0) WRITE (6,238)
35 IF (IU.NE.0) WRITE (6,239)
36 IF (IW.NE.0) WRITE (6,240)
37 IF (IPHIS.NE.0) WRITE (6,241)
38 IF (IPHIT.NE.0) WRITE (6,242)
39 IF (IPHI.NE.0) WRITE (6,243)
40 IF (IPHI.NE.0) WRITE (6,244)
41 IF (IPHI.NE.0) WRITE (6,245)
42 GO TO 6
43 WRITE (6,246) KMAX,MAXM,DELOAD,LSMAX,ITRMAX,EPS
44 WRITE (6,247) CHAR,TKN,ELAST,SIGO,TEEO,NU
45 IF (NTHMAX.EQ.0) GO TO 7
46 WRITE (6,248)
47 WRITE (6,249) (TH(NTH),NTH=1,NTHMAX)
48 IF (NDIMEN.EQ.1) WRITE (6,250)

```

```

00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
0001000
0001010
0001020
0001030
0001040
0001050
0001060
0001070
0001080
0001090
0001100
0001110
0001120
0001130
0001140
0001150
0001160
0001170
0001180
0001190
0001200
0001210
0001220
0001230

```



```

1,REFERENCE THICKNESS-----,E12.4/5X,REFERENCE EL00001720
2ASTICITY-----,E12.4/5X,REFERENCE STRESS-----0001730
3-----,E12.4/5X,POISSONS RATIO-----0001740
4,E12.4/5X,PLOTS HAVE BEEN REQUESTED FOR THE BELOW LISTED GE0001750
214FORMAT (,T10,STIFFNESS OR LOADING QUANTITIES:,,)000031763
1OMETRY, (,T70,RADIUS,,)0001770
215FORMAT (,T70,GAMMA,,)0001780
216FORMAT (,T70,OMEGA-S,,)0001800
217FORMAT (,T70,OMEGA-THETA,,)0001810
218FORMAT (,T70,DEOMEGA-S,,)0001820
219FORMAT (,T70,B-STIFFNESS,,)0001830
220FORMAT (,T70,D-STIFFNESS,,)0001840
221FORMAT (,T70,B-PRIME,,)0001850
222FORMAT (,T70,D-PRIME,,)0001860
223FORMAT (,T70,PRESSURE LOADING - NORMAL,,)0001873
224FORMAT (,T70,PRESSURE LOADING - MERIDIONAL,,)0001880
225FORMAT (,T70,PRESSURE LOADING - CIRCUMFERENTIAL,,)0001890
226FORMAT (,T70,THERMAL LOADING,,)0001900
227FORMAT (,T70,THERMAL MOMENT,,)0001910
228FORMAT (,T70,D-THERMAL LOADING,,)0001923
229FORMAT (,T70,D-THERMAL MOMENT,,)0001930
230FORMAT (,T70,D-THERMAL MOMENT,,)0001940
231FORMAT (,T20,PLOTS HAVE BEEN REQUESTED FOR THE BELOW LISTED MO00001953
1DAL QUANTITIES:,,)000001960
232FORMAT (,T20,PLOTS HAVE BEEN REQUESTED FOR THE BELOW LISTED SU0001970
1MED QUANTITIES:,,)0001983
233FORMAT (,T70,N-S,,)0001990
234FORMAT (,T70,N-THETA,,)0002000
235FORMAT (,T70,N-S-THETA,,)0002010
236FORMAT (,T70,Q-S,,)0002020
237FORMAT (,T70,M-S,,)0002033
238FORMAT (,T70,M-THETA,,)0002040
239FORMAT (,T70,M-S-THETA,,)0002050
240FORMAT (,T70,U,,)0002060
242FORMAT (,T70,W,,)0002070
241FORMAT (,T70,V,,)0002080
243FORMAT (,T70,PHI-S,,)0002090
244FORMAT (,T70,PHI-THETA,,)0002100
245FORMAT (,T70,PHI,,)0002113
246FORMAT (,T4X,NUMBER OF STATIONS-----,I3/5X,N00032113
NUMBER OF MODES-----,E9.4/5X,MAXIMUM NUMBER OF TIME STEPS-----0002120
2-----,E9.4/5X,MAXIMUM NUMBER OF ITERATIONS-----,I3/5X,C00032143
3-----,I5/5X,MAXIMUM NUMBER OF ITERATIONS-----,F6.4/5X0002150
4ONVERGENCE CRITERION-----,E12.4/5X,CHARACTERISTIC SHELL DIMENSION-----000002160
247FORMAT (,T4X,CHARACTERISTIC SHELL DIMENSION-----,E12.4/5X,REFERENCE EL00002170
1,REFERENCE THICKNESS-----,E12.4/5X,REFERENCE STRESS-----00002180
2ASTICITY-----,E12.4/5X,REFERENCE TIME-----0002190
3-----,E12.4/5X,REFERENCE TIME-----0002190

```



```

4,E12.4/5X,'POISSONS RATIO-----',E12.4//)
248 FORMAT (' ',T20,'CIRCUMFERENTIAL COORDINATES FOR THE PRINT RECORD,
1 IN RADIANT MEASURE, ARE: '//)
249 FORMAT (' ',T10,6F16.10)
253 FORMAT ('/// ',T20,'THE DATA PRINTED IS NON-DIMENSIONAL')
251 FORMAT ('/// ',T20,'THE DATA PRINTED IS DIMENSIONAL')
RETURN
END
00002200
00002210
00002220
00002230
00002240
00002250
00002260
00002270

```

```

SUBROUTINE STATIC
***** THIS SUBROUTINE IS THE MAJOR CONTROLLING ROUTINE FOR ALL STATIC
***** ANALYSIS PROBLEMS. IT CONDUCTS INITIALIZATION, PREPARES INPUT
***** MATERIALS FOR PLOTTING IF REQ'D, DETERMINES SHELL GEOMETRY, STIFF-
***** NESSES, LOADING (PHYSICAL AND/OR THERMAL), AND INITIAL CONDITIONS.
***** IT CONTROLS PROBLEM SOLUTION PROCEDURE.
***** IMPLICIT LOGICAL*1 ($)
***** REAL*4 NU,LAM,LAM2,JAY,MT,LSD18,LSDIN,MASS
COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(10),MNINIT
COMMON /IBL3/ MO,M1,M2,M3
COMMON /IBL4/ KMAX,KL
COMMON /IBL5/ IBCINL,IBCFNL
COMMON /IBL6/ KLL
COMMON /IBL7/ MNMAXC,MAXD(10),MAXS(10),MAXSY(10),IS(10,10),
1 JS(10,10),ID(10,10),JD(10,10),IJS(10)
COMMON /IBL8/ LSTEP,ITR
COMMON /IBL9/ MAXM
COMMON /IBL10/ IFREQ,NTHMAX
COMMON /IBL11/ ICORFL,IPASS
COMMON /IBL12/ KMAX1,KMAX2,NCONV
COMMON /IBL13/ ITRMAX,LSMAX
COMMON /BL1/ A(4,4),BEE(4,4),C(4,4)
COMMON /BL3/ PR(10),PX(10),PT(10)
COMMON /BL4/ P(4,4,200),X(4,200),ZF1M(4,4,10),ZF2M(4,4,10),
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280

```



```

4      YNSTH(100),YQS(100),YMS(100),YMTH(100),YMSTH(100),00000770
5      YU(100),YV(100),XSTATN(100),YW(100),YPHIS(100),YPHIT(100),00000780
6      YPHI(100),XSTATN(100),YU(100),YV(100),YW(100),YPHIS(100),YPHIT(100),00000790
COMMON /BLDATA/ TITLE,NO,IMODE,NDIMEN,IPRINT,LCHMAX,IC 00000800
DIMENSION SIGT(2),SIGC(2),TITLE(18) *****00000810
C*****00000820
WRITE (6,8888) *****00000830
DELSD=DELOAD*DELOAD *****00000840
KL=KMAX-1 *****00000850
KLL=KMAX-2 *****00000860
KMAX1=KMAX+1 *****00000870
KMAX2=KMAX+2 *****00000880
AK=KL *****00000890
SIGT(1)=SIGO*TKN *****00000900
SIGT(2)=SIGO/ELAST *****00000910
SIGC(1)=SIGO*CHAR/ELAST *****00000920
SIGC(2)=SIGO*TKN**3/CHAR *****00000930
IF (IBCINL.LT.0) GO TO 14 *****00000940
DO 98 I=1,4 *****00000950
DO 98 J=1,4 *****00000960
KKLM=J/4+1 *****00000970
OMEG1(I,J)=OMEG1(I,J)*SIGT(KKLM) *****00000980
CAPL1(I,J)=CAPL1(I,J)*SIGC(KKLM) *****00000990
IF (IBCFNL.LT.0) GO TO 17 *****00001000
DO 99 I=1,4 *****00001010
DO 99 J=1,4 *****00001020
KKLM=J/4+1 *****00001030
OMEG1(I,J)=OMEG1(I,J)*SIGT(KKLM) *****00001040
CAPL1(I,J)=CAPL1(I,J)*SIGC(KKLM) *****00001050
LAM=TKN/CHAR *****00001060
SOE=SIGO/ELAST *****00001070
OSE=.5*SOE *****00001080
SI=1.0-NU *****00001090
LAM2=LAM**2 *****00001100
IF(NDIMEN.LT.1) GO TO 228 *****00001110
SIGO=1.0 *****00001120
ELAST=1.0 *****00001130
TKN=1.0 *****00001140
CHAR=1.0 *****00001150
DO 230 M=1,MAXM *****00001160
N(M)=0.0 *****00001170
PX(M)=0.0 *****00001180
PT(M)=0.0 *****00001190
PR(M)=0.0 *****00001200
TT(M)=0.0 *****00001210
MT(M)=0.0 *****00001220
DT(M)=0.0 *****00001230
*****00001240

```



```

DMT(M)=0.0
MAXD(M)=0
MAXS(M)=3
MAXSY(M)=0
CALL GEOM
1 ICHCK1=IABS(IGAMMA)+IABS(IOMEGS)+IABS(IOMEGT)+IABS(IDEOMS)
  ICHCK2=IABS(IRADII)+IABS(IDSTIF)+IABS(IBBSTF)+IABS(IDDSTF)
  IF (.NOT.$PLOTS) GO TO 1
DO 2 K=1,KMAX
2 XSTATN(K)=FLOAT(K)
  IF (ICHCK1.EQ.0) GO TO 1
DO 1 K=1,KMAX
  XRADII(K)=R(K)*CHAR
  YGAMMA(K)=GAM(K)/CHAR
  YOMEGS(K)=OMXI(K)/CHAR
  YOMEGT(K)=OMT(K)/CHAR
  YDEOMS(K)=DEOMX(K)/(CHAR*CHAR)
1 CONTINUE
DO 86 K=1,KMAX
86 MASS(K)=0.
  WRITE(6,802)
DO 978 K=1,KMAX
  RKK=R(K)*CHAR
  OMXIK=OMXI(K)/CHAR
  GAMK=GAM(K)/CHAR
  OMTK=OMT(K)/CHAR
  DEOMXK=DEOMX(K)/(CHAR*CHAR)
978 WRITE(6,803) K,RKK,GAMK,OMXIK,OMTK,DEOMXK
805 M0=0
  M1=0
  M2=0
  M3=0
  ABN=CHAR/SIGO/TKN
  ZN=SIGO*TKN
  WRITE(6,112)
DO 888 K=1,KMAX
  CALL BDB(K,B,DB,D,DD)
  BST=ELAST*TKN
  ZST=ELAST*TKN**3
  B=B*BST
  D=D*ZST
  DD=DB/CHAR*BST
  DD=DD/CHAR*ZST
  WRITE(6,71) K,B,D,DB,DD
  IF (.NOT.$PLOTS.OR.(ICHCK2.EQ.0)) GO TO 888
  YBSTIF(K)=B
  YDSTIF(K)=D

```

```

00001250
00001260
00001270
00001280
00001290
00001300
00001310
00001320
00001330
00001340
00001350
00001360
00001370
00001380
00001390
00001400
00001410
00001420
00001430
00001440
00001450
00001460
00001470
00001480
00001490
00001500
00001510
00001520
00001530
00001540
00001550
00001560
00001570
00001580
00001590
00001600
00001610
00001620
00001630
00001640
00001650
00001660
00001670
00001680
00001690
00001700
00001710
00001720

```



```

YBDSIF(K)=DB
YDSSIF(K)=DD
888 CONTINUE
CALL PLOAD(1)
CALL TLOAD(1)
DO 889 M=1,MNMAX
WRITE(6,113) N(M)
WRITE(6,114)
1 ICHCK3=IABS(IPR)+IABS(IPT)+IABS(ITT)+IABS(IMT)
+IABS(IDTT)+IABS(IDMT)
DO 890 K=1,KMAX
CALL PLOAD(K)
CALL TLOAD(K)
PRM=PR(M)/ABN
PTM=PT(M)/ABN
PXM=PX(M)/ABN
TTM=TT(M)*ZN
ETM=MT(M)/CHAR*ZN*TKN*TKN
DTM=DT(M)/CHAR*ZN
DMTM=DMT(M)*ZN*TKN*TKN/(CHAR*CHAR)
WRITE(6,115) K,PRM,PTM,TTM,ETM,DTM,DMTM
IF (.NOT.$PLOTS.OR.(ICHCK3.EQ.0)) GO TO 890
YPR(K)=PRM
YPS(K)=PXM
YPT(K)=PTM
YTI(K)=TTM
YMT(K)=ETM
YDT(K)=DTM
YDMT(K)=DMTM
890 CONTINUE
IF (M.EQ.1) ICHCK3=ICHCK1+ICHCK2+ICHCK3
IF ($PLOTS.AND.(ICHCK3.GT.0)) CALL PLOT1(M)
889 CONTINUE
DELSQ=DELSQ*2
TDL1=.5/DEL
TDEL=2.0*DEL
MNINIT=1
MNMAXO=MNMAX
DO 23 I=1,4
DO 20 J=1,4
UNIT(I,J)=0.0
20 IF(I.EQ.J) UNIT(I,J)=1.0
NMAX=MAX(M*KMAX2
DO 22 K=1,NMAX
DO 22 I=1,4
ZDOT(I,K)=0.0
ZO(I,K)=J.0
Z2(I,K)=0.0

```



```

22  Z3(I,K)=0.0
    Z(I,K)=0.0
    ALOAD=DELOAD
    CALL PMATRIX
    LSTEP=1
    LCHANG=0
    ITR=1
    ICORFL=0
    IF(MNMAX.EQ.MAXM) ICORFL=1
    IPASS=0
    CALL XANDZ
    IF(ITRMAX.EQ.1) GO TO 50
    MNMAXO=MNMAX
    IF(IPASS.LT.2) CALL MODES
    IF((NCONV.EQ.1).AND.(ITR.GT.1)) GO TO 50
    IF(ITR.LT.ITRMAX) GO TO 23
    IF(LCHANG.LT.LCHMAX) GO TO 30
    WRITE(6,220) NO
    GO TO 500
50  FL=LSTEP
    FI=IPRINT
    LI=LSTEP/IPRINT
    FLI=LI
    FT=FLI-FL/FI
    IF(FT.EQ.0.) CALL OUTPUT(IMODE)
    IF(LSTEP.EQ.1) ITR=1
    IF(LSTEP.EQ.1) ITRPR=1
    IF(ITR.GT.ITRPR) ITRPR=ITR
    IF(LSTEP.GE.LSMAX) GO TO 360
60  DO 61 MN=1,MNMAXO
    DO 61 K=1,KMAX2
    IK=K+(MN-1)*KMAX2
    DO 61 I=1,4
    ZN=2.0*Z(I,IK)-ZO(I,IK)
    ZO(I,IK)=Z(I,IK)
    Z(I,IK)=ZN
61  IF(LSTEP.GE.LSMAX) GO TO 360
62  ALOAD=ALOAD+DELOAD
    LSTEP=LSTEP+1
    ITR=1
    GO TO 400
360 WRITE(6,221) NO
    GO TO 500
23  ITR=ITR+1
    GO TO 400
30  IF(LSTEP-1) 310,310,320
310 WRITE(6,223)
    GO TO 500

```

```

00002210
00002220
00002230
00002240
00002250
00002260
00002270
00002280
00002290
00002300
00002310
00002320
00002330
00002340
00002350
00002360
00002370
00002380
00002390
00002400
00002410
00002420
00002430
00002440
00002450
00002460
00002470
00002480
00002490
00002500
00002510
00002520
00002530
00002540
00002550
00002560
00002570
00002580
00002590
00002600
00002610
00002620
00002630
00002640
00002650
00002660
00002670
00002680

```



```

320 WRITE(6,222)
    LCHANG=LCHANG+1
    LSTEP=LSTEP-1
    ALOAD=ALOAD-DELOAD
    DELOAD=DELOAD/5.0
    DO 32 MN=1,MNMAX0
    DO 32 K=1,KMAX2
    IK=K+(MN-1)*KMAX2
    DO 32 I=1,4
    Z(I,IK)=Z(I,IK)
32 GO TO 62
C*****
71 FORMAT(20X,I3,4X,4E20.6)
112 IFNESS(20H,12H STATION 20H B STIFFNESS 20H D ST000028200
113 IFNESS(20H,12H STATION 20H B PRIME D PRIME 20H D ST000028300
114 FORMAT(5X,7HSTATION,3X,15H PR 15H MT 15H PX 15H DTT 100002870
15H PT 15H DMT 100002880
115 FORMAT(6X,I3,7X,7E15.4)
220 FORMAT(1H,80H THE MAXIMUM NUMBER OF LOAD CHANGES HAVE BEEN 00002890
1EN MADE. END PROBLEM NUMBERI4) 00002900
221 FORMAT(1H,79H THE MAXIMUM NUMBER OF LOAD STEPS HAVE BEEN 00002910
1 TAKEN. END PROBLEM NUMBERI4) 00002920
222 FORMAT(1H,119H THE SOLUTION DID NOT CONVERGE WITHIN THE M00002940
1AXIMUM NUMBER OF ITERATIONS. THE LOAD FACTOR HAS BEEN DIVIDED BY 00002950
25.) 00002960
223 FORMAT(1H,69H THE SOLUTION DID NOT CONVERGE FOR THE FIRS00002970
1T LOAD INCREMENT./11X,71HLOOK FOR AN ERROR IN THE INPUT DATA, OR T00002980
2RY A SMALLER VALUE FOR DELOAD.) 00002990
802 FORMAT(1H,17X,15H STATION 16H RADIUS 16H DEOMEGA S 16H GAMMA 00003000
16H OMEGA S 16H THETA16H 00003010
888 FORMAT(20X,I3,9X,5E16.4) 00003020
500 FORMAT('0',I20,'EXECUTING IN SUBROUTINE "STATIC"') 00003030
END 00003040
00003050

```



```

SUBROUTINE DYNAMC
C*****
C THIS SUBROUTINE IS ONE OF THE MAJOR CONTROLLING SUBROUTINES FOR
C ALL DYNAMIC ANALYSIS PROBLEMS. IT OPERATES IN A FASHION SIMILAR
C TO SUBROUTINE STATIC.
C*****
C IMPLICIT LOGICAL*1 ($)
C REAL*4 NU,LAM,LAM2,JAY,MT, LSD18,LSD1N,MASS
COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(10), MNINIT
C
COMMON /IBL3/ MO,M1,M2,M3
COMMON /IBL4/ KMAX,KL
COMMON /IBL5/ IBCINL, IBCFNL
COMMON /IBL6/ KLL
COMMON /IBL7/ MNMAXO, MAXD(10), MAXS(10), MAXSY(10), IS(10,10),
1 JS(10,10), ID(10,10), JD(10,10), IJS(10)
COMMON /IBL8/ LSTEP, ITR
COMMON /IBL9/ MAXM
COMMON /IBL10/ IFREQ, NTHMAX
COMMON /IBL11/ ICORFL, IPASS
COMMON /IBL12/ KMAX1, KMAX2, NCONV
COMMON /IBL13/ ITRMAX, LSMAX
COMMON /BL1/ A(4,4), BEE(4,4), C(4,4)
COMMON /BL3/ PR(10), PX(10), PT(10)
COMMON /BL4/ P(4,4,200), X(4,200), ZF1M(4,4,10), ZF2M(4,4,10),
1 ZF3M(4,4,10), ZF4M(4,4,10)
COMMON /BL5/ TT(10), MT(10), DT(10), DMT(10)
C
COMMON /BL6/ Z(4,220), SOE, OSE, ALOAD
COMMON /BL7/ D1, S1
COMMON /BL8/ R(200), GAM(200), DMT(200)
COMMON /BL9/ FFS(4,10), ELIS(4), GEES(4,10)
COMMON /BL10/ PHIX(10), PHIT(10), PHI(10)
COMMON /BL11/ OMXI(200), PHEE, TO, T2
COMMON /BL12/ TDLI, TDEL
COMMON /BL13/ OMEGI(4,4), CAPL1(4,4), OMEGL(4,4), CAPLL(4,4),
1 UNIT(4,4)
COMMON /BL14/ LAM2, LSD18, LSD1N
COMMON /BL15/ NU, UL(10), V1(10), W1(10), V2(10), U2(10), W2(10), U3(10),
1 V3(10), W3(10)
COMMON /BL16/ EPS
COMMON /BL17/ DEL
COMMON /BL18/ ELI(4), ELL(4)
COMMON /BL19/ TH(36)
COMMON /BL20/ DEOMX(200)
COMMON /BL23/ JAY(4,4), H(4,4)
COMMON /BL24/ DL(4,4,10), DG(4,4,10), DF(4,4,10)

```



```

COMMON /BL25/ E(4,4),F(4,4),G(4,4)
COMMON /BL27/ BX3(10),BT3(10),BXT3(10),BE3(10)
COMMON /BL28/ EXX3(10),ETX3(10),ETX3(10),EX3(10),ET3(10)
COMMON /BL29/ BX1(10),BT1(10),BXT1(10),BE1(10),BX2(10),BT2(10),
1 BX12(10),BE2(10)
COMMON /BL30/ EXX1(10),ETX1(10),ETX1(10),EX1(10),ET1(10),EXX2(10),
1 ETX2(10),ETX2(10),EX2(10),ET2(10)
COMMON /BL31/ DELSQ,EXT1(10)
COMMON /BL32/ TKN,ELAST,CHAR,SIGO
COMMON /BL34/ DEE(4,4,200),DST(4,4,200)
COMMON /BL100/ TEOO,$DYNMC
COMMON /BL101/ ZQ(4,220),Z2(4,220),Z3(4,220),DELSQ
COMMON /BL102/ DELOAD
COMMON /BL103/ MASS(200)
COMMON /BL104/ ZDOT(4,220)
COMMON /BL110/ TX(10),TTH(10),TX(10),MX(10),MTH(10),MXT(10),
1 QS(10)
COMMON /BL111/ ABZ,ABZO,ABZN,ABZ3,DD2
COMMON /BLPLOT/ IRADII,IGAMMA,ICOMEGS,IOMEGT,IDEOMS,IBSTIF,IDSTIF,
1 IBBSTF,IDDSTF,IPR,IPS,IPT,ITT,IMT,IDTT,IDMT,INS,
2 INTH,INSTH,IQS,IMS,IMTH,IMSTH,IU,IV,IW,IPHIS,
3 IPHIT,IPHI,$PLOTS,$MODAL
COMMON /BLPLT1/ XRADII(100),YGAMMA(100),YOMEGS(100),YOMEGT(100),
1 YDEOMS(100),YBSTIF(100),YDSTIF(100),YBBSTF(100),
2 YDDSTF(100),YPR(100),YPS(100),YPT(100),YTT(100),
3 YMT(100),YDIT(100),YDMT(100),YNS(100),YNTH(100),
4 YNSTH(100),YQS(100),YMS(100),YMTTH(100),YMSSTH(100),
5 YU(100),YV(100),YW(100),YPHIS(100),YPHIT(100),
6 YPHI(100),XSTATN(100)
COMMON /BLDATA/ TITLE,NO,IMODE,NDIMEN,IPRINT,LCHMAX,IC
DIMENSION SIGT(2),SIGC(2),TITLE(18)
DIMENSION VBAR(10),AVB(10)
C*****
WRITE (6,888)
DELSQ=DELOAD*DELOAD
KL=KMAX-1
KLL=KMAX-2
KMAX1=KMAX+1
KMAX2=KMAX+2
AK=KL
SIGT(1)=SIGO*TKN
SIGT(2)=SIGO/ELAST
SIGC(1)=SIGO*CHAR/ELAST
SIGC(2)=SIGO*TKN**3/CHAR
IF (IBCNL.LT.0) GO TO 14
DO 98 I=1,4
DO 98 J=1,4
KKLM=J/4+1
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960

```



```

98 OMEGL(I,J)=OMEG1(I,J)*SIGT(KKLM)
14 CAPL1(I,J)=CAPL1(I,J)*SIGC(KKLM)
  IF(IBCFLN.LT.0) GO TO 17
  DO 99 I=1,4
  DO 99 J=1,4
  KKLM=J/4+1
  OMEGL(I,J)=OMEG1(I,J)*SIGT(KKLM)
99 CAPL1(I,J)=CAPL1(I,J)*SIGC(KKLM)
17 LAM=TKN/CHAR
  SOE=SIGO/ELAST
  OSE=.5*SOE
  DI=1.0-NU
  SI=1.0+NU
  LAM2=LAM**2
  IF(NDIMEN.LT.1) GO TO 228
  SIGO=1.0
  ELAST=1.0
  TKN=1.0
  CHAR=1.0
228 DO 230 M=1,MAXM
  N(M)=0.0
  PX(M)=0.0
  PT(M)=0.0
  PR(M)=0.0
  TT(M)=0.0
  MT(M)=0.0
  DT(M)=0.0
  DMT(M)=0.0
  MAXD(M)=0
  MAXS(M)=0
  MAXSY(M)=0
230 CALL GEOM
  ICHK1=IABS(IGAMMA)+IABS(IOMEGS)+IABS(IOMEGT)+IABS(IDEOMS)
  ICHK2=IABS(IRADII)+IABS(ISTIF)+IABS(ISTIF)+IABS(ISTIF)
1  IF (.NOT.$PLOTS) GO TO 1
  DO 2 K=1,KMAX
  XSTATN(K)=FLOAT(K)
2  IF (ICCHK1.EQ.0) GO TO 1
  DO 1 K=1,KMAX
  XRADI(K)=R(K)*CHAR
  YGAMMA(K)=GAM(K)/CHAR
  YOMEGS(K)=OMX(K)/CHAR
  YOMEGT(K)=OMT(K)/CHAR
  YDEOMS(K)=DEOMX(K)/(CHAR*CHAR)
1  CONTINUE
804 WRITE(6,810)
  TEED=TEEO

```



```

IF(NDIMEN.EQ.1) TEED=1.0
DO 979 K=1,KMAX
  RKX=R(K)*CHAR
  OMXIK=OMXI(K)/CHAR
  GAMK=GAM(K)/CHAR
  OMTK=OMT(K)/CHAR
  DEOMXK=DEOMX(K)/(CHAR*CHAR)
  AMSS=MASS(K)*TEED**2*ELAST*TKN/CHAR**2
979 WRITE(6,813) K,RKK,GAMK,OMXIK,OMTK,DEOMXK,AMSS
805 MO=0
  M1=0
  M2=0
  M3=0
  ABN=CHAR/SIGO/TKN
  ZN=SIGC*TKN
  WRITE(6,112)
DO 888 K=1,KMAX
  CALL BDB(K,B,DB,D,DD)
  BST=ELAST*TKN
  ZST=ELAST*TKN**3
  B=B*BST
  D=D*ZST
  DB=DB/CHAR*BST
  DD=DD/CHAR*ZST
  WRITE(6,71) K,B,D,DB,DD
  IF(.NOT.$PLOTS.OR.(ICHECK2.EQ.0)) GO TO 888
  YBSTIF(K)=B
  YDSTIF(K)=D
  YBBSTIF(K)=DB
  YDDSTIF(K)=DD
  CONTINUE
888 CALL PLOAD(1)
  CALL TLOAD(1)
  DELSQ=DEL**2
  TDLI=.5/DEL
  TDEL=2.0*DEL
  MNINIT=J
  MNMAXO=MNMAX
DO 20 I=1,4
DO 20 J=1,4
  UNIT(I,J)=0.0
20 IF(I.EQ.J) UNIT(I,J)=1.0
  NMAX=MAXM*KMAX2
DO 22 K=1,NMAX
DO 22 I=1,4
  ZDOT(I,K)=0.0
  ZO(I,K)=0.0
  Z2(I,K)=0.0

```

```

00001450
00001460
00001470
00001480
00001490
00001500
00001510
00001520
00001530
00001540
00001550
00001560
00001570
00001580
00001590
00001600
00001610
00001620
00001630
00001640
00001650
00001660
00001670
00001680
00001690
00001700
00001710
00001720
00001730
00001740
00001750
00001760
00001770
00001780
00001790
00001800
00001810
00001820
00001830
00001840
00001850
00001860
00001870
00001880
00001890
00001900
00001910
00001920

```


00001930
00001940
00001950
00001960
00001970
00001980
00001990
00002000
00002010
00002020
00002030
00002040
00002050
00002060
00002070
00002080
00002090
00002100
00002110
00002120
00002130
00002140
00002150
00002160
00002170
00002180
00002190
00002200
00002210
00002220
00002230
00002240
00002250
00002260
00002270
00002280
00002290
00002300
00002310
00002320
00002330
00002340
00002350
00002360
00002370
00002380
00002390
00002400

```

22      Z3(I,K)=0.0
        Z(I,K)=0.0
        IF(IC.EQ.0) GO TO 834
        CALL INITL
        ACM=CHAR*SIGO/ELAST
        ACM=SIGO*TKN**3/CHAR
        DO 830 M=1,MNMAX
          MM=(M-1)*KMAX2
          WRITE(6,126) N(M)
          WRITE(6,127)
          DO 831 K=2,KMAX1
            MK=K+MM
            TU=ACD*ZO(1,MK)
            TV=ACD*ZO(2,MK)
            TW=ACD*ZO(3,MK)
            TM=ACM*ZO(4,MK)
            KK=K-1
            WRITE(6,71) KK,TU,TV,TW,TM
831      WRITE(6,129)
          DO 830 K=2,KMAX1
            ACM=CHAR*SIGO/(ELAST*TEEO)
            AMD=SIGO*TKN**3/(CHAR*TEEO)
            MK=K+MM
            TU=ACD*ZDOT(1,MK)
            TV=ACD*ZDOT(2,MK)
            TW=ACD*ZDOT(3,MK)
            TM=AMD*ZDOT(4,MK)
            KK=K-1
            WRITE(6,71) KK,TU,TV,TW,TM
          DO 830 I=1,4
            Z(I,MK)=ZO(I,MK)+ZDOT(I,MK)*DELOAD
            Z2(I,MK)=ZO(I,MK)-ZDOT(I,MK)*DELOAD
830      Z3(I,MK)=ZO(I,MK)-2.*ZDOT(I,MK)*DELOAD
834      CONTINUE
          ALOAD=1.0
          CALL PMATRIX
          LSTEP=1
          LCHANG=0
          ITR=1
          ICORFL=0
          IF(MNMAX.EQ.MAXM) ICORFL=1
          IPASS=0
          CALL XANDZ
          IF(ITRMAX.EQ.1) GO TO 50
          MNMAXO=MNMAX
          IF(IPASS.LT.2) CALL MODES
          IF(NCONV.EQ.1) GO TO 50
          IF(ITR.LT.ITRMAX) GO TO 23
400

```



```

1 AT TIME STEPI5,21H. END PROBLEM NUMBERI4,1H.)
271 FORMAT(1H,79H THE MAXIMUM NUMBER OF TIME STEPS HAVE BEEN
1 TAKEN. END PROBLEM NUMBERI4)
273 FORMAT(1H,69H THE SOLUTION DID NOT CONVERGE FOR THE FIRST
IT TIME INCREMENT./11X,71HLOOK FOR AN ERROR IN THE INPUT DATA, OR
TRY A SMALLER VALUE FOR DELOAD.)
810 FORMAT(1H, 5X,15H STATION 16H RADIUS 16H GAMMA
1 16H OMEGA S 16H DEOMEGA S 16H
2 MASS
813 FORMAT( 8X,I3, 9X,6E16.4)
8888 FORMAT (',',T10,'EXECUTING IN SUBROUTINE "DYNAMIC"')
500 RETURN
END
00002890
0000002900
00002910
0000002920
0000002930
00002940
00002950
00002960
00002970
00002980
00002990
00003000
00003010

```

```

SUBROUTINE OUTPUT(I MODE)
THIS SUBROUTINE PREPARES THE PRINTOUT MATERIAL. EVERY IPRINT *****
CONVERGED SOLUTION IS PRINTED. THE FOURIER COEFFICIENTS OF THE *****
IN PLANE FORCES, MERIDIONAL TRANSVERSE FORCE, CIRCUMFERENTIAL *****
BENDING MOMENT, TWISTING MOMENT AND ROTATIONS CAN BE COMPUTED *****
AND PRINTED WITH THE SOLUTION Z FOR THE FOURIER COEFFICIENTS *****
OF THE THREE DISPLACEMENTS AND MERIDIONAL BENDING MOMENT. THIS *****
OUTPUT MATERIAL IS CONVERTED FROM DIMENSIONLESS FORM TO DIMEN *****
SIONAL FORM HERE. 1. 2IFREQ+1, ETC. THIS SUBROUTINE ALSO PERFORMS *****
SUMMATION PROCESS FOR COMPUTING THE TOTAL VALUES OF THE FORCES *****
MOMENTS, DISPLACEMENTS AND ROTATIONS AT THE NTHMAX POSITIONS *****
AROUND THE CIRCUMFERENCE PRESCRIBED IN THE INPUT DATA. *****
DATA TO BE PLOTTED IS PREPARED HERE, IF REQ'D *****
***** IMPLICIT LOGICAL*1 ($) *****
REAL NU, MT, MX, MTH, MXT, MTS, KX, KT, KXT, LAM, LAM2, MASS *****
COMMON /IBL2/ N(10), MNINIT *****
COMMON /IBL3/ M3, M1, M2, M3 *****
COMMON /IBL4/ KMAX, KL *****
COMMON /IBL5/ IBCINL, IBCFNL *****
COMMON /IBL7/ MNMAXO, MAXD(10), MAXS(10), MAXSY(10), IS(10,10), *****
00000010 *****
00000020 *****
00000030 *****
00000040 *****
00000050 *****
00000060 *****
00000070 *****
00000080 *****
00000090 *****
00000100 *****
00000110 *****
00000120 *****
00000130 *****
00000140 *****
00000150 *****
00000160 *****
00000170 *****
00000180 *****
00000190 *****
00000200 *****
00000210 *****
00000220 *****
00000230 *****

```



```

1 COMMON /IBL8/ JS(10,10), ID(10,10), JD(10,10), IJS(10)
COMMON /IBL10/ LSTEP, ITR
COMMON /IBL12/ IFRFREQ, NTHMAX
COMMON /IBL13/ KMAX1, KMAX2, NCONV
COMMON /IBL4/ ITRMAX, LSMAK
COMMON /BL4/ P(4,4,200), X(4,200), ZF1M(4,4,10), ZF2M(4,4,10),
1 ZF3M(4,4,10), ZF4M(4,4,10)
COMMON /BL5/ ZT(10), MT(10), DT(10), DMT(10)
COMMON /BL6/ Z(4,220), SOE, OSE, ALOAD
COMMON /BL7/ D1, S1
COMMON /BL8/ R(200), GAM(200), DMT(200)
COMMON /BL10/ PHIX(10), PHIT(10), PHI(10)
COMMON /BL11/ OMXI(200), PHEE, TO, T2
COMMON /BL12/ TDL1, TDEL
COMMON /BL14/ LAM2, LSD18, LSD1N
COMMON /BL15/ NU, UI(10), VI(10), W1(10), V2(10), U2(10), W2(10), U3(10),
1 V3(10), W3(10)
COMMON /BL17/ DEL
COMMON /BL19/ TH(36)
COMMON /BL20/ DEOMX(200)
COMMON /BL27/ BX3(10), BT3(10), BXT3(10), BE3(10)
COMMON /BL31/ DELSQ, EXT1(10)
COMMON /BL32/ TKN, ELAST, CHAR, SIGO
COMMON /BL100/ TEO, $DYNMC
COMMON /BL101/ ZO(4,220), Z2(4,220), Z3(4,220), DELSD
COMMON /BL102/ DELOAD
COMMON /BL103/ MASS(200)
COMMON /BL110/ TX(10), TTH(10), TXT(10), MX(10), MTH(10), MXT(10),
1 QS(10)
COMMON /BL111/ ABZ, ABZ0, ABZN, ABZ3, DD2
COMMON /BLPLOT/ IRADII, IGAMMA, IOMEGS, IOMEGT, IDEOMS, IBST IF, IDSTIF,
1 IBBSTF, IDSTF, IPR, IPS, IPT, ITT, IMT, IDTT, IDMT, INS,
2 INTHT, INSTH, IQS, ITS, IMTH, IMSTH, IU, IV, IW, IPHIS,
3 IPHIT, IPHI, $PLOTS, $MODAL
COMMON /BLPLT1/ XRADII(100), YGAMMA(100), YOMEGS(100), YOMEGT(100),
1 YDEOMS(100), YBSTIF(100), YDSTIF(100), YBBSTIF(100),
2 YDDSTF(100), YPR(100), YPS(100), YPT(100), YTT(100),
3 YMT(100), YDTT(100), YDMT(100), YNS(100), YNTH(100),
4 YNSTH(100), YQS(100), YMS(100), YMTH(100), YMSTH(100),
5 YU(100), YV(100), YW(100), YPHIS(100), YPHIT(100),
6 YPHI(100), XSTATN(100)
DIMENSION PTF(200), PF(200)
1 *****
2 ABZO=SIGO/ELAST
3 IF ($DYNMC) GO TO 181
4 WRITE(6,101) LSTEP, ALOAD, ITR
5 GO TO 182
6 TI=LSTEP*DELOAD
181

```



```

DTI=TI*TEEO
WRITE(6,151) LSTEP, TI, DTI, ITR
LAM=TKN/CHAR
ENL=1.
ABZ=1.
ABZ3=ABZ*TKN*TKN/CHAR
ABZN=CHAR*SIGJ/ELAST
IF(1TRMAX.EQ.1) ENL=0.
DD2=1.-NU**2
D2I=1./DD2
DPI=1./SI
DNI=1./DI
TDLSQL=.5/DELSQ
1 ICHCK1=IABS(INS)+IABS(INTH)+IABS(IQS)+IABS(IMS)
+IABS(IMTH)+IABS(IMSTH)
1 ICHCK2=IABS(IU)+IABS(IV)+IABS(IW)+IABS(IPHIS)+IABS(IPHIT)
+IABS(IPHI)
1 IF(NTHMAX.EQ.0) GO TO 991
DO 21 NTH=1, NTHMAX
DO 1 MN=1, MNMAXO
I1=1+(MN-1)*KMAX2
I2=I1+1
U1(MN)=Z(1, I1)
U2(MN)=Z(1, I2)
V1(MN)=Z(2, I1)
V2(MN)=Z(2, I2)
W1(MN)=Z(3, I1)
W2(MN)=Z(3, I2)
1 THET=TH(NTH)
WRITE(6,116) THET
DO 121 K=1, KMAX
K1=K+1
CALL BDB(K, BS, DB, DS, DD)
IF(K.EQ.1.AND.IBCINL.LT.0) CALL POLE(K)
IF(K.EQ.1.AND.IBCINL.LT.0) GO TO 999
IF(K.EQ.KMAX.AND.IBCFNL.LT.0) CALL POLE(K)
IF(K.EQ.KMAX.AND.IBCFNL.LT.0) GO TO 999
CALL PHIBET(K)
DEX=DEOMX(K)
RRA=1./R(K)
OX=OMXI(K)
OT=OMT(K)
GA=GAM(K)
DOXT=OX-OT
GDO=GA*DOXT
DD2D=DD2*DS
DO 3 MN=1, MNMAXO
EN=N(MN)

```

182

1



```

ENR=EN*RRRA
CALL TLOAD(K)
TTS=TT(MN)*ALOAD
EX=(U3(MN)-U1(MN))*TDLI+OX*W2(MN)+ENL*OSE*(BX3(MN)+BE3(MN))
ET=ENR*V2(MN)+GA*U2(MN)+OT*W2(MN)+ENL*OSE*(BT3(MN)+BE3(MN))
EXT=.5*((V3(MN)-V1(MN))*TDLI-ENR*U2(MN)-GA*V2(MN)+ENL*SOE*BX3(MN)
1)
KT=ENR*PHIT(MN)+GA*PHIX(MN)
KXT=.5*(ENR*(-PHIX(MN)-GA*W2(MN)+(W3(MN)-W1(MN))*TDLI)+GDO*V2(MN)
+OT*(V3(MN)-V1(MN))*TDLI-GA*PHIT(MN)-DOXT*PHI(MN))
1
TX(MN)=BS*(EX+NU*ET)-TTS
TTH(MN)=BS*(ET+NU*EX)-TTS
TXT(MN)=BS*D1*EXT
MK1=K1+(MN-1)*KMAX2
MX(MN)=Z(4,MK1)
MTH(MN)=NU*MX(MN)+DD2D*KT-D1*MT(MN)*ALOAD
MXT(MN)=DS*D1*KXT
MK11=MK1+1
MK1=MK1-1
QS(MN)=SIGO*TKN*LAM2*(GA*MX(MN)+(Z(4,MK11)-Z(4,MK1))*TDLI
1
+ENR*MXT(MN)-GA*MTH(MN))
MX(MN)=MX(MN)*ABZ3
MTH(MN)=MTH(MN)*ABZ3
MXT(MN)=MXT(MN)*ABZ3
TX(MN)=TX(MN)*ABZ
TTH(MN)=TTH(MN)*ABZ
TXT(MN)=TXT(MN)*ABZ
PHIX(MN)=PHIX(MN)*ABZO
PHIT(MN)=PHIT(MN)*ABZO
PHI(MN)=PHI(MN)*ABZO
U1(MN)=U2(MN)
U2(MN)=U3(MN)
V1(MN)=V2(MN)
V2(MN)=V3(MN)
W1(MN)=W2(MN)
W2(MN)=W3(MN)
FK=K-1
3
FIFREQ=IFREQ
KTSI=(K-1)/IFREQ
FKTST=KTSI
FKTEST=FK/FIFREQ-FKTST
IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 999
IF(FKTEST.NE.0.) GO TO 2
999
X(1,K)=0.
X(2,K)=0.
X(3,K)=0.
X(4,K)=0.
PTF(K)=0.

```


00001680
00001690
00001700
00001710
00001720
00001730
00001740
00001750
00001760
00001770
00001780
00001790
00001800
00001810
00001820
00001830
00001840
00001850
00001860
00001870
00001880
00001890
00001900
00001910
00001920
00001930
00001940
00001950
00001951
00001952
00001953
00001954
00001955
00001960
00001970
00001980
00001990
00002000
00002010
00002020
00002030
00002040
00002050
00002060
00002070
00002080
00002090
00002100

```

PF(K)=0.
AMX=0.
AMTH=0.
AMXTH=0.
ANX=0.
ANTH=0.
ANXTH=0.
AQS=0.
DO 72 MN=1,MNMAXO
EN=N(MN)
FC=EN*THET
SN=SIN(FC)
CS=COS(FC)
X(1,K)=X(1,K)+U1(MN)*CS*ABZN
X(2,K)=X(2,K)+V1(MN)*SN*ABZN
X(3,K)=X(3,K)+W1(MN)*CS*ABZN
X(4,K)=X(4,K)+PHIX(MN)*CS
PTF(K)=PTF(K)+PHIT(MN)*SN
AMX=AMX+MX(MN)*CS
AMTH=AMTH+MTH(MN)*CS
AMXTH=AMXTH+MX T(MN)*SN
ANX=ANX+TX(MN)*CS
ANTH=ANTH+TTH(MN)*CS
AQS=AQS+QS(MN)*CS
ANXTH=ANXTH+TX T(MN)*SN
PF(K)=PF(K)+PHI(MN)*SN
CONTINUE
72
429 IF(K.EQ.1) WRITE(6,117)
IF((K.EQ.1.AND.IBCINL.LT.0).OR.(K.EQ.KMAX.AND.IBCFNL.LT.0))
1 GO TO 221
GO TO 221
220 WRITE (6,1181) K,ANX,ANTH,ANXTH,AMX,AMTH,AMXTH
GO TO 1354
221 WRITE(6,118) K,ANX,ANTH,ANXTH,AQS,AMX,AMTH,AMXTH
1354 IF ($MODAL.OR.(ICHECK1.EQ.0)) GO TO 2
YNS(K)=ANX
YNTH(K)=ANTH
YNSTH(K)=ANXTH
YQS(K)=AQS
YMS(K)=AMX
YMTH(K)=AMTH
YMSTH(K)=AMXTH
CONTINUE
2
121 CONTINUE
DO 663 K=1,KMAX
FK=K-1
FIFREQ=1FREQ
KTST=(K-1)/1FREQ

```



```

FKTST=KFTST
FKTEST=FK/FIFREQ-FKST
IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 661
IF(FKTEST.NE.0.) GO TO 658
IF(K.EQ.1) WRITE(6,217)
661 WRITE(6,218) K,X(1,K),X(2,K),X(3,K),X(4,K),PTF(K),PF(K)
IF ($MODAL.OR.(ICHCK2.EQ.0)) GO TO 658
YU(K)=X(1,K)
YV(K)=X(2,K)
YW(K)=X(3,K)
YPHIS(K)=X(4,K)
YPHIT(K)=PTF(K)
YPHI(K)=PF(K)
DO 659 I=1,4
658 X(I,K)=0.
660 CONTINUE
IF ($PLOTS.AND..NOT.$MODAL.AND.((ICHCK1.GT.0).OR.(ICHCK2.GT.0)))
1 CALL PLOT2(NTH)
21 CONTINUE
991 IF(IMODE.LE.0) RETURN
DO 534 MN=1,MNMAXO
WRITE(6,749) N(MN)
DO 521 MM=1,MNMAXO
I1=1+(MM-1)*KMAX2
I2=I1+1
U1(MM)=Z(1,I1)
U2(MM)=Z(1,I2)
V1(MM)=Z(2,I1)
V2(MM)=Z(2,I2)
W1(MM)=Z(3,I1)
W2(MM)=Z(3,I2)
521 CONTINUE
DO 445 K=1, KMAX
K1=K+1
CALL BDB(K,BS,DB,DS,DD)
IF(K.EQ.1.AND.IBCINL.LT.0) CALL POLE(K)
IF(K.EQ.KMAX.AND.IBCFNL.LT.0) CALL POLE(K)
IXZ=TX(MN)
ITHZ=TH(MN)
ITXZ=TX(MN)
ITXZ=TX(MN)
AMXZ=MX(MN)
AMTHZ=TH(MN)
AMXTZ=MX(MN)
QSZ=QS(MN)
X(1,K)=PHIX(MN)
X(2,K)=PHIT(MN)
X(3,K)=PHI(MN)
IF(K.EQ.1.AND.IBCINL.LT.0) GO TO 583

```



```

IF(K.EQ.KMAX.AND.IBCFNL.LT.0) GO TO 583
CALL PHIBET(K)
DEX=DEOMX(K)
RRA=1./R(K)
OX=OMXI(K)
OT=OMT(K)
GA=GAM(K)
DOXT=OX-OT
GDO=GA*DOXT
DD2D=DD2*DS
EN=N(MN)
ENR=EN*RRA
CALL TLOAD(K)
TTS=TT(MN)*ALJAD
EX=(U3(MN)-U1(MN))*TDLI+OX*W2(MN)+ENL*OSE*(BX3(MN)+BE3(MN))
ET=ENR*V2(MN)+GA*U2(MN)+OT*W2(MN)+ENL*OSE*(BT3(MN)+BE3(MN))
EXT=.5*(V3(MN)-V1(MN))*TDLI-ENR*U2(MN)-GA*V2(MN)+ENL*SOE*BXT3(MN)
1) KT=ENR*PHIT(MN)+GA*PHIX(MN)
KXT=.5*(ENR*(-PHIX(MN)-GA*W2(MN)+(W3(MN)-W1(MN))*TDLI)+GDO*V2(MN)
+OT*(V3(MN)-V1(MN))*TDLI-GA*PHIT(MN)-DOXT*PHI(MN))
1 TXZ=(BS*(EX+NU*ET))-TTS)*ABZ
TTHZ=(BS*(ET+NU*EX))-TTS)*ABZ
TXTZ=BS*DI*EXT*ABZ
MKI=K1+(MN-1)*KMAX2
AMXZ=Z(4,MK1)
AMTHZ=NU*AMXZ+DD2D*KT-D1*MT(MN)*ALOAD
AMXTZ=DS*D1*KXT
MKI1=MKI+1
MKK1=MKI-1
1 QSZ=SIGO*TKN*LAM2*(GA*AMXZ+(Z(4,MK11)-Z(4,MKK1))*TDLI+ENR*AMXTZ
-GA*AMTHZ)
AMXZ=AMXZ*ABZ3
AMTHZ=AMTHZ*ABZ3
AMXTZ=AMXTZ*ABZ3
X(1,K)=PHIX(MN)*ABZO
X(2,K)=PHIT(MN)*ABZO
X(3,K)=PHI(MN)*ABZO
DO 533 MM=1,MNMAXO
U1(MM)=U2(MM)
U2(MM)=U3(MM)
V1(MM)=V2(MM)
V2(MM)=V3(MM)
W1(MM)=W2(MM)
W2(MM)=W3(MM)
FK=K-1
FIFREQ=IFREQ
KTST=(K-1)/IFREQ
533
00002590
00002600
00002610
00002620
00002630
00002640
00002650
00002660
00002670
00002680
00002690
00002700
00002710
00002720
00002730
00002740
00002750
00002760
00002770
00002780
00002790
00002800
00002810
00002820
00002830
00002840
00002850
00002860
00002870
00002880
00002890
00002900
00002910
00002920
00002930
00002940
00002950
00002960
00002970
00002980
00002990
00003000
00003010
00003020
00003030
00003040
00003050
00003060

```



```

FKTST=KTST
FKTEST=FK/FIFREQ-FKTST
IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 583
IF(FKTEST.NE.0.) GO TO 445
583 CONTINUE
IF(K.EQ.1) WRITE(6,117)
IF((K.EQ.1.AND.IBCINL.LT.0).OR.(K.EQ.KMAX.AND.IBCFNL.LT.0))
1 GO TO 4444
GO TO 4445
4444 WRITE(6,1181) K, TXZ, TTHZ, TXTZ, AMXZ, AMTHZ, AMXTZ
GO TO 4446
4445 WRITE(6,118) K, TXZ, TTHZ, TXTZ, QSZ, AMXZ, AMTHZ, AMXTZ
4446 IF(.NOT.$PLOTS.OR..NOT.$MODAL.OR.(ICHECK1.EQ.0)) GO TO 445
YNS(K)=TXZ
YNTH(K)=TTHZ
YNSTH(K)=TXTZ
YQS(K)=QSZ
YMS(K)=AMXZ
YMTH(K)=AMTHZ
YMSTH(K)=AMXTZ
CONTINUE
445 WRITE(6,217)
446 DO 447 K=1,KMAX
FK=K-1
FIFREQ=IFREQ
KTST=(K-1)/IFREQ
FKTST=KTST
FKTEST=FK/FIFREQ-FKTST
IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 593
IF(FKTEST.NE.0.) GO TO 446
593 KZ=K+1+(MN-1)*KMAX2
UP=Z(1,KZ)*ABZN
VP=Z(2,KZ)*ABZN
WP=Z(3,KZ)*ABZN
WRITE(6,218) K, UP, VP, WP, X(1,K), X(2,K), X(3,K)
IF(.NOT.$PLOTS.OR..NOT.$MODAL.OR.(ICHECK2.EQ.0)) GO TO 447
YU(K)=UP
YV(K)=VP
YW(K)=WP
YPHIS(K)=X(1,K)
YPHIT(K)=X(2,K)
YPHI(K)=X(3,K)
CONTINUE
447 IF ($PLOTS.AND.$MODAL.AND.((ICHECK1.GT.0).OR.(ICHECK2.GT.0)))
1 CALL PLOT2(1)
534 CONTINUE
C*****
101 FORMAT('1',
THE LOAD STEP NUMBER IS '12',
THE L00003490

```



```

C*****6*****YPHI(100),XSTATN(100)*****00000190
NGKMAX=-KMAX*****00000191
IF ($MODAL) GO TO 121
IF (INS.EQ.0) GO TO 4
WRITE (6,1000)
IF (INS.GT.0) CALL PLOTIT (XSTATN,YNS,KMAX,0)
IF (INS.LT.0) CALL PLOTIT (XSTATN,YNS,NGKMAX,0)
WRITE (6,1001) TH(NTH)
IF (INTH.EQ.0) GO TO 5
4 WRITE (6,1000)
IF (INTH.GT.0) CALL PLOTIT (XSTATN,YNTH,KMAX,0)
IF (INTH.LT.0) CALL PLOTIT (XSTATN,YNTH,NGKMAX,0)
WRITE (6,1002) TH(NTH)
5 IF (INSTH.EQ.0) GO TO 6
WRITE (6,1000)
IF (INSTH.GT.0) CALL PLOTIT (XSTATN,YNSTH,KMAX,0)
IF (INSTH.LT.0) CALL PLOTIT (XSTATN,YNSTH,NGKMAX,0)
WRITE (6,1003) TH(NTH)
6 IF (IQS.EQ.0) GO TO 7
WRITE (6,1000)
IF (IQS.GT.0) CALL PLOTIT (XSTATN,YQS,KMAX,0)
IF (IQS.LT.0) CALL PLOTIT (XSTATN,YQS,NGKMAX,0)
WRITE (6,1004) TH(NTH)
7 IF (IMS.EQ.0) GO TO 8
WRITE (6,1000)
IF (IMS.GT.0) CALL PLOTIT (XSTATN,YMS,KMAX,0)
IF (IMS.LT.0) CALL PLOTIT (XSTATN,YMS,NGKMAX,0)
WRITE (6,1005) TH(NTH)
8 IF (IMTH.EQ.0) GO TO 9
WRITE (6,1000)
IF (IMTH.GT.0) CALL PLOTIT (XSTATN,YMTH,KMAX,0)
IF (IMTH.LT.0) CALL PLOTIT (XSTATN,YMTH,NGKMAX,0)
WRITE (6,1006) TH(NTH)
9 IF (IMSTH.EQ.0) GO TO 1211
WRITE (6,1000)
IF (IMSTH.GT.0) CALL PLOTIT (XSTATN,YMSTH,KMAX,0)
IF (IMSTH.LT.0) CALL PLOTIT (XSTATN,YMSTH,NGKMAX,0)
WRITE (6,1007) TH(NTH)
1211 IF (IU.EQ.0) GO TO 10
WRITE (6,1000)
IF (IU.GT.0) CALL PLOTIT (XSTATN,YU,KMAX,0)
IF (IU.LT.0) CALL PLOTIT (XSTATN,YU,NGKMAX,0)
WRITE (6,1010) TH(NTH)
10 IF (IV.EQ.0) GO TO 11
WRITE (6,1000)
IF (IV.GT.0) CALL PLOTIT (XSTATN,YV,KMAX,0)
IF (IV.LT.0) CALL PLOTIT (XSTATN,YV,NGKMAX,0)
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650

```



```

11 WRITE (6,1009) TH(NTH)
   IF (IW.EQ.0) GO TO 12
   WRITE (6,1030)
   IF (IW.GT.0) CALL PLOTIT (XSTATN,YW,KMAX,0)
   IF (IW.LT.0) CALL PLOTIT (XSTATN,YW,NGKMAX,0)
   WRITE (6,1008) TH(NTH)
12 IF (IPHS.EQ.0) GO TO 13
   WRITE (6,1000)
   IF (IPHS.GT.0) CALL PLOTIT (XSTATN,YPHIS,KMAX,0)
   IF (IPHS.LT.0) CALL PLOTIT (XSTATN,YPHIS,NGKMAX,0)
   WRITE (6,1011) TH(NTH)
13 IF (IPHIT.EQ.0) GO TO 14
   WRITE (6,1000)
   IF (IPHIT.GT.0) CALL PLOTIT (XSTATN,YPHIT,KMAX,0)
   IF (IPHIT.LT.0) CALL PLOTIT (XSTATN,YPHIT,NGKMAX,0)
   WRITE (6,1012) TH(NTH)
14 IF (IPHI.EQ.0) GO TO 21
   WRITE (6,1000)
   IF (IPHI.GT.0) CALL PLOTIT (XSTATN,YPHI ,KMAX,0)
   IF (IPHI.LT.0) CALL PLOTIT (XSTATN,YPHI ,NGKMAX,0)
   WRITE (6,1013) TH(NTH)
21 RETURN
121 IF (INS.EQ.0) GO TO 15
   WRITE (6,1000)
   IF (INS.GT.0) CALL PLOTIT (XSTATN,YNS,KMAX,0)
   IF (INS.LT.0) CALL PLOTIT (XSTATN,YNS,NGKMAX,0)
   WRITE (6,2001)
15 IF (INTH.EQ.0) GO TO 16
   WRITE (6,1000)
   IF (INTH.GT.0) CALL PLOTIT (XSTATN,YNTH,KMAX,0)
   IF (INTH.LT.0) CALL PLOTIT (XSTATN,YNTH,NGKMAX,0)
   WRITE (6,2002)
16 IF (INSTH.EQ.0) GO TO 17
   WRITE (6,1000)
   IF (INSTH.GT.0) CALL PLOTIT (XSTATN,YNSTH,KMAX,0)
   IF (INSTH.LT.0) CALL PLOTIT (XSTATN,YNSTH,NGKMAX,0)
   WRITE (6,2003)
17 IF (IQS.EQ.0) GO TO 18
   WRITE (6,1000)
   IF (IQS.GT.0) CALL PLOTIT (XSTATN,YQS,KMAX,0)
   IF (IQS.LT.0) CALL PLOTIT (XSTATN,YQS,NGKMAX,0)
   WRITE (6,2004)
18 IF (IMS.EQ.0) GO TO 19
   WRITE (6,1000)
   IF (IMS.GT.0) CALL PLOTIT (XSTATN,YMS,KMAX,0)
   IF (IMS.LT.0) CALL PLOTIT (XSTATN,YMS,NGKMAX,0)
   WRITE (6,2005)
19 IF (IMTH.EQ.0) GO TO 22

```

```

00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
00010000
00010010
00010020
00010030
00010040
00010050
00010060
00010070
00010080
00010090
00010100
00010110
00010120
00010130

```



```

WRITE (6,1000)
IF (IMTH.GT.0) CALL PLOTIT (XSTATN,YMTH,KMAX,0)
IF (IMTH.LT.0) CALL PLOTIT (XSTATN,YMTH,NGKMAX,0)
WRITE (6,2006)
IF (IMSTH.EQ.0) GO TO 231
WRITE (6,1000)
IF (IMSTH.GT.0) CALL PLOTIT (XSTATN,YMSTH,KMAX,0)
IF (IMSTH.LT.0) CALL PLOTIT (XSTATN,YMSTH,NGKMAX,0)
WRITE (6,2007)
IF (IU.EQ.0) GO TO 23
WRITE (6,1000)
IF (IU.GT.0) CALL PLOTIT (XSTATN,YU,KMAX,0)
IF (IU.LT.0) CALL PLOTIT (XSTATN,YU,NGKMAX,0)
WRITE (6,2010)
IF (IV.EQ.0) GO TO 24
WRITE (6,1000)
IF (IV.GT.0) CALL PLOTIT (XSTATN,YV,KMAX,0)
IF (IV.LT.0) CALL PLOTIT (XSTATN,YV,NGKMAX,0)
WRITE (6,2009)
IF (IW.EQ.0) GO TO 25
WRITE (6,1000)
IF (IW.GT.0) CALL PLOTIT (XSTATN,YW,KMAX,0)
IF (IW.LT.0) CALL PLOTIT (XSTATN,YW,NGKMAX,0)
WRITE (6,2008)
IF (IPHIS.EQ.0) GO TO 26
WRITE (6,1000)
IF (IPHIS.GT.0) CALL PLOTIT (XSTATN,YPHIS,KMAX,0)
IF (IPHIS.LT.0) CALL PLOTIT (XSTATN,YPHIS,NGKMAX,0)
WRITE (6,2011)
IF (IPHIT.EQ.0) GO TO 27
WRITE (6,1000)
IF (IPHIT.GT.0) CALL PLOTIT (XSTATN,YPHIT,KMAX,0)
IF (IPHIT.LT.0) CALL PLOTIT (XSTATN,YPHIT,NGKMAX,0)
WRITE (6,2012)
IF (IPHI.EQ.0) GO TO 28
WRITE (6,1000)
IF (IPHI.GT.0) CALL PLOTIT (XSTATN,YPHI,KMAX,0)
IF (IPHI.LT.0) CALL PLOTIT (XSTATN,YPHI,NGKMAX,0)
WRITE (6,2013)
28 RETURN
C*****
1000 FORMAT ('1')
1001 FORMAT ('0',T10,'S"-MEMBRANE FORCE VS STATN, MERIDIAN AT THETA =',F10.5,' RADIANS')
1002 FORMAT ('0',T10,'THETA"-MEMBRANE FORCE VS STATN, MERIDIAN AT THE',F10.5,' RADIANS')
1003 FORMAT ('0',T10,'S-THETA" MEMBRANE FORCE VS STATN, MERIDIAN AT T',F10.5,' RADIANS')
1 THETA =

```

```

00001140
00001150
00001160
00001170
00001180
00001190
00001200
00001210
00001220
00001230
00001240
00001250
00001260
00001270
00001280
00001290
00001300
00001310
00001320
00001330
00001340
00001350
00001360
00001370
00001380
00001390
00001400
00001410
00001420
00001430
00001440
00001450
00001460
00001470
00001480
00001490
00001500
00001510
00001520
00001530
00001540
00001550
00001560
00001570
00001580
00001590
00001600
00001610

```



```

SUBROUTINE FLYNVY
WRITE (6,11)
WRITE (6,2)
WRITE (6,3)
WRITE (6,4)
WRITE (6,5)
WRITE (6,6)
WRITE (6,7)
WRITE (6,8)
WRITE (6,9)
WRITE (6,10)
WRITE (6,11)
WRITE (6,12)
WRITE (6,13)
WRITE (6,14)
WRITE (6,15)
WRITE (6,16)
WRITE (6,17)
WRITE (6,18)
WRITE (6,19)
WRITE (6,20)
WRITE (6,21)
WRITE (6,22)
WRITE (6,23)
WRITE (6,24)
WRITE (6,25)
WRITE (6,26)
WRITE (6,27)
WRITE (6,28)
WRITE (6,29)
WRITE (6,30)
WRITE (6,31)
WRITE (6,32)
WRITE (6,33)
WRITE (6,34)
WRITE (6,35)
WRITE (6,36)
WRITE (6,37)
WRITE (6,38)
WRITE (6,39)
WRITE (6,40)
111 FORMAT ('1',T11,'
111 FORMAT (XX,')
211 FORMAT ('1',T11,'
311 FORMAT (XXX,')
311 FORMAT ('1',T11,'

```

```

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480

```



```

1  XXX  XXX')
4  FORMAT (' ',T11,'
1  XXX  XXX')
5  FORMAT (' ',T11,'
1  FORMAT (X@XXX')
6  FORMAT (' ',T11,'XXXXXXXXXXXXXXXXXXXXXXXXX
1  XXXXXXXXXX
2  XX')
7  FORMAT (' ',T11,'XXXXXXXXXX
1  XXXXXXXXXX
2)
8  FORMAT (' ',T11,'
1  FORMAT (XXX
9  FORMAT (' ',T11,'
1  XXXXXXXX
10  FORMAT (' ',T11,'
1  XXXXXXXX
11  FORMAT (' ',T11,'
1  XXXXXXXX
12  FORMAT (' ',T11,'
1  XXXXXXXX
13  FORMAT (' ',T11,'
1  XXXXXXXX
14  FORMAT (' ',T11,'
1  XXXXXXXX
15  FORMAT (' ',T11,'
1  XXXXXXXX
16  FORMAT (' ',T11,'
1  XXXXXXXX
17  FORMAT (' ',T11,'
1  XXXXXXXX
18  FORMAT (' ',T11,'
1  XXXXXXXX
19  FORMAT (' ',T11,'
20  FORMAT (' ',T11,'
21  FORMAT (' ',T11,'
22  FORMAT (' ',T11,'
23  FORMAT (' ',T11,'
24  FORMAT (' ',T11,'
25  FORMAT (' ',T11,'
26  FORMAT (' ',T11,'

```



```

1 00000000' }
27 1 FORMAT (' ',T11,'
      000')
28 1 FORMAT (' ',T11,'
      000')
29 1 FORMAT (' ',T11,'
      000')
30 1 2)
      1 2)
      2)
31 1 2)
      1 2)
      2)
32 1 2)
      1 2)
      2)
33 1 2)
      1 2)
      2)
34 1 2)
      1 2)
      2)
35 1 2)
      1 2)
      2)
36 1 2)
      1 2)
      2)
37 1 2)
      1 2)
      2)
38 1 2)
      1 2)
      2)
39 1 2)
      1 2)
      2)
40 1 2)
      1 2)
      2)
      RETURN
      END

```



```

SUBROUTINE FORCE(K)
C*****
C THIS SUBROUTINE COMPUTES THE GEE VECTOR IN EQUATION (28) OF
C REF. 3, AND THE X VECTOR IN EQUATION (29A) OF REF. 3 FOR A GIVEN MER-
C IDIONAL STATION K. THE VECTOR 'GEES' IS THE NONLINEAR VALUE OF 'GEE'
C AT STATION 1.
C*****
IMPLICIT LOGICAL*1 ($)
REAL NU,MT,LAM2,MASS,MAS
COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(10),MNINIT
COMMON /IBL4/ KMAX,KL
COMMON /IBL8/ LSTEP,ITR
COMMON /IBL12/ KMAX1,KMAX2,NCONV
COMMON /IBL13/ ITRMAX,LSMAX
COMMON /BL3/ PR(10),PX(10),PT(10)
COMMON /BL4/ P(4,4,200),X(4,200),ZF1M(4,4,10),ZF2M(4,4,10),
1 ZF3M(4,4,10),ZF4M(4,4,10)
COMMON /BL5/ TT(10),MT(10),DT(10),DMT(10)
COMMON /BL6/ Z(4,220),SOE,OSE,ALOAD
COMMON /BL7/ D1,S1
COMMON /BL8/ R(200),GAM(200),OMT(200)
COMMON /BL9/ FFS(4,10),ELIS(4),GEES(4,10)
COMMON /BL11/ OMX1(200),PHEE,T0,T2
COMMON /BL12/ TDLI,TDEL
COMMON /BL14/ LAM2,LSDL8,LSDDIN
COMMON /BL15/ NU,U1(10),V1(10),W1(10),V2(10),U2(10),W2(10),U3(10),
1 V3(10),W3(10)
COMMON /BL17/ DEL
COMMON /BL24/ DL(4,4,10),DG(4,4,10),DF(4,4,10)
COMMON /BL27/ BX3(10),BT3(10),BXT3(10),BE3(10)
COMMON /BL28/ EXX3(10),ETT3(10),ETX3(10),EXT3(10),EX3(10),ET3(10)
COMMON /BL29/ BX1(10),BT1(10),BXT1(10),BE1(10),BX2(10),BT2(10),
1 BXT2(10),BE2(10)
COMMON /BL30/ EXX1(10),ETT1(10),ETX1(10),EXT1(10),EX1(10),ET1(10),EXX2(10),
1 ETT2(10),ETT2(10),EXT2(10),EX2(10),ET2(10)
COMMON /BL31/ DELSQ,EXT1(10)
COMMON /BL34/ DEE(4,4,200),DST(4,4,200)
COMMON /BL100/ TEEQ,$DYNMC
COMMON /BL101/ ZO(4,220),Z2(4,220),Z3(4,220),DELSO
COMMON /BL102/ DELOAD
COMMON /BL103/ MASS(200)
DIMENSION GEE(4)
C*****
FDIFF(A,B,C)=(-1.5*A+2.*B-.5*C)/DEL
RS=R(K)
RR=1./RS
GA=GAM(K)
C*****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470

```



```

OX=OMXI(K)
OT=OMT(K)
DL2=D1*LAM2
CALL BDB(K,BS,DBS,D,DD)
CALL PLOAD(K)
CALL TLOAD(K)
MAS=MASS(K)
DO 4 M=1,MNMAX
  IZ=K+1+(M-1)*KMAX2
  IK=K+(M-1)*KMAX
  IK1=IK-1
  EN=N(M)
  ENR=EN*RR
  EMT=MT(M)
  GEE(1) = (-PX(M)+DT(M)-DL2*GA*OX*EMT)*TDEL*ALOAD
1  + MAS*(-5.*ZO(1,IZ)+4.*Z2(1,IZ)-Z3(1,IZ))*TDEL/DELS
1  GEE(2) = (-PT(M)-ENR*TT(M)-DL2*ENR*OT*EMT)*TDEL*ALOAD
1  + MAS*(-5.*ZO(2,IZ)+4.*Z2(2,IZ)-Z3(2,IZ))*TDEL/DELS
1  GEE(3) = (-PR(M)-(OX+OT)*TT(M)-DL2*(GA*DMT(M)-(OX*OT-ENR**2)
1  *MT(M)))*TDEL*ALOAD
1  + MAS*(-5.*ZO(3,IZ)+4.*Z2(3,IZ)-Z3(3,IZ))*TDEL/DELS
1  GEE(4) = MT(M)*TDEL*ALOAD
  IF(1)TRMAX.EQ.1) GO TO 50
  IF(K.GT.1) GO TO 6
  BX2T=BX1(M)
  BT2T=BT1(M)
  BX12T=BX11(M)
  BE2T=BE1(M)
  EXX2T=EXX1(M)
  ET2T=ETT1(M)
  ETX2T=ETX1(M)
  EX2T=EX1(M)
  ET2T=ET1(M)
  DBX= FDIFF(BX2T,BX2(M),BX3(M))
  DBT= FDIFF(BT2T,BT2(M),BT3(M))
  DBE= FDIFF(BX12T,BX12(M),BX13(M))
  DET= FDIFF(BE2T,BE2(M),BE3(M))
  DEX= FDIFF(ET2T,ET2(M),ET3(M))
  DEXX= FDIFF(EX2T,EX2(M),EXX3(M))
  DETX= FDIFF(ETX2T,ETX2(M),ETX3(M))
  GO TO 7
6 IF(K.LT.KMAX) GO TO 8
  BX2T=BX3(M)
  BT2T=BT3(M)
  BX12T=BX13(M)
  BE2T=BE3(M)

```


00000960
00000970
00000980
00000990
00001000
00001010
00001020
00001030
00001040
00001050
00001060
00001070
00001080
00001090
00001100
00001110
00001120
00001130
00001140
00001150
00001160
00001170
00001180
00001190
00001200
00001210
00001220
00001230
00001240
00001250
00001260
00001270
00001280
00001290
00001300
00001310
00001320
00001330
00001340
00001350
00001360
00001370
00001380
00001390
00001400
00001410
00001420
00001430

```

EXX2T=EXX3(M)
ETT2T=ETT3(M)
ETX2T=ETX3(M)
EXX2T=EXX3(M)
ET2T=ET3(M)
DBX=-FDIFF(BX2T,BX2(M),BX1(M))
DBT=-FDIFF(BT2T,BT2(M),BT1(M))
DBXT=-FDIFF(BXT2T,BXT2(M),BXT1(M))
DBE=-FDIFF(BE2T,BE2(M),BE1(M))
DET=-FDIFF(ET2T,ET2(M),ET1(M))
DEX=-FDIFF(EX2T,EX2(M),EX1(M))
DEXX=-FDIFF(EXX2T,EXX2(M),EXX1(M))
DET2T=-FDIFF(ETX2T,ETX2(M),ETX1(M))
GO TO 7
DBX=TDLI*(BX3(M)-BX1(M))
DBT=TDLI*(BT3(M)-BT1(M))
DBE=TDLI*(BE3(M)-BE1(M))
DET=TDLI*(ET3(M)-ET1(M))
DEX=TDLI*(EX3(M)-EX1(M))
DEXX=TDLI*(EXX3(M)-EXX1(M))
DET2T=TDLI*(ETX3(M)-ETX1(M))
BX2T=BX2(M)
BT2T=BT2(M)
BE2T=BE2(M)
ET2T=ET2(M)
EX2T=EX2(M)
ET2T=ET2(M)
GEE(1)=GEE(1)+ENR*D1*OSE*(BS*(DBX+DBE+GA*D1*(BX2T-BT2T)+NU*(DBT+DBE)
+ENR*D1*BX2T)+DBS*(BX2T+BE2T+NU*(BT2T+BE2T))-2.*OX*
(ETX2T+ETX2T)-ENR*(ETX2T+ETX2T)*TDEL
GEE(2)=GEE(2)+OSE*(BS*(DBX+DBE+GA*D1*(BX2T-BT2T)+NU*(DBT+DBE)
+ENR*D1*BX2T)+DBS*(BX2T+BE2T+NU*(BT2T+BE2T))-D1*
(ETX2T+ETX2T)-ENR*(ETX2T+ETX2T)*TDEL
GEE(3)=GEE(3)+OSE*(BS*(DBX+DBE+GA*D1*(BX2T-BT2T)+NU*(DBT+DBE)
+ENR*D1*BX2T)+DBS*(BX2T+BE2T+NU*(BT2T+BE2T))-D1*
(ETX2T+ETX2T)-ENR*(ETX2T+ETX2T)*TDEL
IF(K.GT.1) GO TO 10
IF(M.GT.1) EL1S(1)=0.0
DO 20 I=1,4
GEES(I,M)=GEE(I)
SUMX=0.
DO 21 J=1,4

```



```

C*****
C FOLLOWING CARD CAUSES A SPECIFIED BOUNDARY CONDITION VALUE TO
C EXIST ONLY FOR MODE 'O'.
C*****
C IF (M.NE.1) ELIS(J)=0
21 SUMX=SUMX+DL(I,J,M)*ELIS(J)+DG(I,J,M)*GEE(J)+DF(I,J,M)*FFS(J,M)
20 X(I,IK)=SUMX
10 DO 11 I=1,4
    SUMX=0.
    DO 12 J=1,4
        SUMX=SUMX+DEE(I,J,IK)*GEE(J)-DST(I,J,IK)*X(J,IK1)
12 SUMX=SUMX+DEE(I,J,IK)*GEE(J)-DST(I,J,IK)*X(J,IK1)
11 X(I,IK)=SUMX
4 CONTINUE
RETURN
END
C*****
00001440
00001450
00001460
00001470
00001480
00001490
00001500
00001510
00001520
00001530
00001540
00001550
00001560
00001570
00001580
00001590

```

```

SUBROUTINE MATINV(A,N,B,M,DETERM,IPIVOT,INDEX,NMAX,ISCALE)
C*****
C THIS SUBROUTINE SOLVES THE MATRIX EQUATION AX=B, WHERE A IS
C A SQUARE COEFFICIENT MATRIX AND B IS A MATRIX OF CONSTANT VEC-
C TORS.
C A(INVERSE) IS ALSO OBTAINED AND THE DETERMINANT OF A IS AVAIL-
C ABLE.
C THE FOLLOWING MUST BE DIMENSIONED IN THE CALLING PROGRAM:
C IPIVOT(N MAX), INDEX(N MAX,2), A(N MAX,N MAX), B(N MAX,N MAX)
C WHERE:
C A = NAME OF 2-DIMENSIONAL ARRAY TO BE INVERTED
C N = ORDER OF A - 1<=N<=NMAX
C B = NAME OF 2-DIMENSIONAL ARRAY TO BE MULTIPLIED
C BY A(INVERSE)
C M = NUMBER OF COLUMN VECTORS IN B
C NOTE: M = 0 SIGNALS INVERSION ONLY)
C IPIVOT = TEMPORARY STORAGE BLOCK
C INDEX = TEMPORARY STORAGE BLOCK
C NMAX = MAXIMUM ORDER OF A (AS DIMENSIONED IN THE
C CALLING PROGRAM)
C*****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200

```


250	B(ICOL, L)=SWAP	00000690
260	INDEX(I, 1)=IROW	00000700
270	INDEX(I, 2)=ICOL	00000710
310	PIVOT=A(ICOL, ICOL)	00000720
		00000730
		00000740
	SCALE THE DETERMINANT	00000750
		00000760
1000	PIVOT I=PIVOT	00000770
1005	IF(ABS(DETERM)-R1) 1030, 1010, 1010	00000780
1010	DETERM=DETERM/R1	00000790
	ISCALE=ISCALE+1	00000800
1020	IF(ABS(DETERM)-R1) 1060, 1020, 1020	00000810
	DETERM=DETERM/R1	00000820
	ISCALE=ISCALE+1	00000830
	GO TO 1060	00000840
1030	IF(ABS(DETERM)-R2) 1040, 1040, 1060	00000850
1040	DETERM=DETERM*R1	00000860
	ISCALE=ISCALE-1	00000870
1050	IF(ABS(DETERM)-R2) 1050, 1050, 1060	00000880
	DETERM=DETERM*R1	00000890
	ISCALE=ISCALE-1	00000900
1060	IF(ABS(PIVOT I)-R1) 1090, 1070, 1070	00000910
1070	PIVOT I=PIVOT I/R1	00000920
	ISCALE=ISCALE+1	00000930
1080	IF(ABS(PIVOT I)-R1) 320, 1080, 1080	00000940
	PIVOT I=PIVOT I/R1	00000950
	ISCALE=ISCALE+1	00000960
1090	IF(ABS(PIVOT I)-R2) 2000, 2000, 320	00000970
2000	PIVOT I=PIVOT I*R1	00000980
	ISCALE=ISCALE-1	00000990
2010	IF(ABS(PIVOT I)-R2) 2010, 2010, 320	00001000
	PIVOT I=PIVOT I*R1	00001010
	ISCALE=ISCALE-1	00001020
320	DETERM=DETERM*PIVOT I	00001030
		00001040
		00001050
	DIVIDE PIVOT ROW BY PIVOT ELEMENT	00001060
		00001070
330	A(ICOL, ICOL)=1.0	00001080
340	DO 350 L=1, N	00001090
350	A(ICOL, L)=A(ICOL, L)/PIVOT	00001100
355	IF(M) 380, 380, 360	00001110
360	DO 370 L=1, M	00001120
370	B(ICOL, L)=B(ICOL, L)/PIVOT	00001130
		00001140
	REDUCE NON-PIVOT ROWS	00001150
		00001160
380	DO 550 L1=1, N	


```

390 IF(L1-ICOLUM) 400, 550, 400
400 T=A(L1,ICOLUM)
420 A(L1,ICOLUM)=0.0
430 DO 450 L=1,N
450 A(L1,L)=A(L1,L)-A(ICOLUM,L)*T
455 IF(M) 550, 550, 460
460 DO 500 L=1,M
500 B(L1,L)=B(L1,L)-B(ICOLUM,L)*T
550 CONTINUE
C
C
C      INTERCHANGE COLUMNS
600 DO 710 I=1,N
610 L=N+1-I
620 IF (INDEX(L,1)-INDEX(L,2)) 630, 710, 630
630 JROW=INDEX(L,1)
640 JCOLUM=INDEX(L,2)
650 DO 705 K=1,N
660 SWAP=A(K,JROW)
670 A(K,JROW)=A(K,JCOLUM)
700 A(K,JCOLUM)=SWAP
705 CONTINUE
710 CONTINUE
740 RETURN
      END

```

```

00001170
00001180
00001190
00001200
00001210
00001220
00001230
00001240
00001250
00001260
00001270
00001280
00001290
00001300
00001310
00001320
00001330
00001340
00001350
00001360
00001370
00001380
00001390
00001400
00001410

```

```

C ***** SUBROUTINE ABC *****
C ***** THIS SUBROUTINE COMPUTES THE ELEMENTS OF THE A, BEE, AND C *****
C ***** MATRICES *****
C ***** COMMON /BL1/ A(4,4),BEE(4,4),C(4,4) *****
C ***** COMMON /BL12/ TDLI,TDEL *****
C ***** COMMON /BL17/ DEL *****
C ***** COMMON /BL25/ E(4,4),F(4,4),G(4,4) *****
C ***** D2=2./DEL *****

```

```

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110

```



```

00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200

```

```

DO 1 I=1,4
DO 1 J=1,4
DEIJ=D2*E(I,J)
FIJ=F(I,J)
BEE(I,J)=-2.*DEIJ+TDEL*G(I,J)
C(I,J)=DEIJ-FIJ
1 A(I,J)=DEIJ+FIJ
RETURN
END

```

```

SUBROUTINE PANDD(K,MN)
C*****
C THIS SUBROUTINE COMPUTES THE ELEMENTS OF THE P, P-BAR, AND
C P-HAT MATRICES FOR EACH MEDIAN STATION K AND FOURIER MODE MN
C THESE MATRICES ARE COMPUTED AND SAVED BECAUSE THEY DO NOT
C CHANGE DURING EITHER THE ITERATION PROCEDURE OR THE LOAD INCRE-
C MENT PROCEDURE - AS THEY ARE A FUNCTION OF THE SHELL'S INITIAL
C GEOMETRY AND STIFFNESS.
C*****
COMMON /IBL4/ KMAX,KL
COMMON /BL1/ A(4,4,200),X(4,200),ZF1M(4,4,10),
COMMON /BL4/ P(4,4,200),ZF4M(4,4,10)
1 COMMON /BL34/ DEE(4,4,200),DST(4,4,200)
DIMENSION TM(4,4),IPIVOT(4),INDEX(4,2),X2(4)
C*****
IKL=K+KMAX*(MN-1)
KLI=IKL-1
DO 1 I=1,4
DO 1 J=1,4
SUM=0.
DO 2 L=1,4
SUM=SUM+C(I,L)*P(L,J,KLI)
2 1 TM(I,J)=BEE(I,J)-SUM
CALL MATINV(TM,4,X2,0,DETERM,IPIVOT,INDEX,4,ISCALE)
DO 5 I=1,4
DO 5 J=1,4
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470

```



```

SUMA=0.
SUMC=0.
DO 6 L=1,4
SUMA=SUMA+TM(I,L)*A(L,J)
SUMC=SUMC+TM(I,L)*C(L,J)
6 P(I,J,IKL)=SUMA
5 DEE(I,J,IKL)=TM(I,J)
DST(I,J,IKL)=SUMC
RETURN
END

```

170


```

C      THE LOCATION OF THE INDEX AND THE VARIABLE MAXSY STORES THE
C      TOTAL NUMBER OF SUCH COMBINATIONS.
C      WITH THIS PROCEDURE, THE SERIES OF PRODUCTS THAT MAKE UP THE
C      BETA'S AND AETA'S CONTAIN NO ZERO TERM, AND THE SUMMATION IS
C      CARRIED OUT IN PHIBET(K) AND TEAETA(K) OVER SPECIFICALLY DE-
C      FINED LIMITS.
C*****
COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(10), MNINIT
COMMON /IBL7/ MNMAXO, MAXD(10), MAXS(10), MAXSY(10), IS(10,10),
1      JS(10,10), ID(10,10), JD(10,10), IJS(10)
COMMON /IBL9/ MAXM
COMMON /IBL11/ ICORFL, IPASS
C*****
IF(MAXM.EQ.1) RETURN
IF(MNINIT.GT.MAXM) RETURN
DO 1 MN=1, MNMAXO
NMN=N(MN)
NNS=MN
IF(MNINIT.GT.MN) NNS=MNINIT
DO 1 MM=NNS, MNMAXO
NMN=N(MM)
NTEST=IABS(NMN-NMM)
DO 2 MMFT=1, MNMAX
IF(NTEST.EQ.N(MMFT)) GO TO 10
2 CONTINUE
IF(ICORFL.EQ.1) GO TO 1
MNMAX=MNMAX+1
N(MNMAX)=NTEST
MMFT=MNMAX
IF(MNMAX.EQ.MAXM) ICORFL=1
10 IF(NMN-NMM) 11,1,12
11 LOCD=MAXD(MMFT)+1
MAXD(MMFT)=LOCD
ID(LOCD,MMFT)=MM
JD(LOCD,MMFT)=MN
GO TO 1
12 LOCD=MAXD(MMFT)+1
MAXD(MMFT)=LOCD
ID(LOCD,MMFT)=MN
JD(LOCD,MMFT)=MM
1 CONTINUE
DO 301 MN=1, MNMAXO
NMN=N(MN)
NNS=MN
IF(MNINIT.GT.MN) NNS=MNINIT
DO 301 MM=NNS, MNMAXO
NMM=N(MM)

```



```

NTEST=NMN+NMN
DO 302 MMFT=1, MNMAX
IF(NTEST.EQ.N(MMFT)) GO TO 310
CONTINUE
IF(ICORFL.EQ.1) GO TO 301
IF(MNMAX.GE.MAXM) GO TO 301
MNMAX=MNMAX+1
N(MNMAX)=NTEST
MMFT=MNMAX
IF(MNMAX.GE.MAXM) ICORFL=1
IF(NMN.EQ.NMM) GO TO 360
MAXS=MAXS(MMFT)+1
LOCS(MMFT)=LOCS
IS(LOCS,MMFT)=MN
JS(LOCS,MMFT)=MM
GO TO 301
360 IF(NMN.EQ.0) GO TO 301
MAXSY(MMFT)=1
IJS(MMFT)=MN
CONTINUE
MNINIT=MNMAXO+1
IF(ICORFL.GT.0) IPASS=IPASS+1
IF(IPASS.LT.2.AND.MNINIT.LE.MNMAX) CALL PMATRIX
RETURN
END

```

```

00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990

```

```

SUBROUTINE HJ(K,MN)
C*****
C      THIS SUBROUTINE COMPUTES THE ELEMENTS OF THE H AND JAY
C      MATRICES FOR BOTH BOUNDARIES OF THE SHELL
C*****
REAL L2,LAM2,LSDIN,LSD18,JAY,NU
COMMON /IBL2/ N(1),MNINIT
COMMON /IBL4/ KMAX,KL
COMMON /BL8/ R(200),GAM(200),OMT(200)
COMMON /BL11/ OMXI(200),PHEE,T0,T2
COMMON /BL14/ LAM2,LSD18,LSDIN

```

```

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110

```



```

COMMON /BL15/ NU,U1(10),V1(10),W1(10),V2(10),W2(10),U3(10),
1      COMMON /BL17/ DEL
COMMON /BL20/ DEOMX(200)
COMMON /BL23/ JAY(4,4),H(4,4)
EQUIVALENCE(L2,LAM2)
*****
CALL BDB(K,B,DB,D,DD)
YAH=1.
IF(K.EQ.1.OR.K.EQ.KMAX)YAH=2.
DI=(1.-NU)
GA=GAM(K)
OX=OMXI(K)
RA=R(K)
EN=N(MN)
ENR=EN/RA
REG=0.
IF(YAH.EQ.2.) REG=1.
OT=OMT(K)
OXT=3.*OMXI(K)-OMT(K)
OTX=3.*OMT(K)-OMXI(K)
DL=D*L2*D1*ENR
H(1,1)=B
H(1,2)=0.
H(1,3)=0.
H(1,4)=0.
H(2,1)=0.
H(2,2)=8*D1/2.+L2*D*D1/8.*OTX**2*REG
H(2,3)=DL/2.*OTX*REG
H(2,4)=0.
H(3,1)=0.
H(3,2)=DL*OTX*YAH/4.
ENR2=ENR**2
H(3,3)=L2*D*D1*(YAH*ENR2+(1.+NU)*GA**2)
GA2=GA**2
H(3,4)=L2
H(4,1)=0.
H(4,2)=0.
H(4,3)=-1.
H(4,4)=0.
JAY(1,1)=NU*GA*B
JAY(1,2)=NU*B*ENR
JAY(1,3)=B*(OX+NU*OT)
JAY(1,4)=0.
JAY(2,1)=-B*D1*ENR/2.-DL/8.*OXT*OTX*REG
JAY(2,2)=-GA*H(2,2)
JAY(2,3)=-GA*H(2,3)
JAY(2,4)=0.
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590

```



```

00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720

```

```

JAY(3,1)=-L2*D*D1*((1.+NU)*GA2*OX+ENR2/4.*OXT*YAH)
JAY(3,2)=-GA*DL/2.*(2.*OT*(1.+NU)+OTX/2.*YAH)
JAY(3,3)=-L2*D*D1*((1.+NU+YAH)*GA*ENR2
JAY(3,4)=L2*D*D1*GA
JAY(4,1)=OX
JAY(4,2)=0.
JAY(4,3)=0.
JAY(4,4)=0.
DO 1 I=1,4
DO 1 J=1,4
1 H(I,J)=H(I,J)/2./DEL
RETURN
END

```

```

SUBROUTINE POLE(K)
C *****
C THIS SUBROUTINE PRINTS THE SOLUTION AT AN INITIAL AND A FINAL
C POLE.
C *****
IMPLICIT LOGICAL*1 ($)
REAL NU,MT,MX,MTH,MXT,MTS,KX,KT,KXT,LAM,LAM2,MASS
COMMON /IBL2/ N(10),MNINIT
COMMON /IBL3/ MO,M1,M2,M3
COMMON /IBL4/ KMAX,KL
COMMON /IBL5/ IBCINL,IBCFNL
COMMON /IBL7/ MNMAXO,MAXD(10),MAXS(10),MAXSY(10),IS(10,10),
1 JS(10,10),ID(10,10),JD(10,10),IJS(10,
COMMON /IBL8/ LSTEP,ITR
COMMON /IBL10/ IFREQ,NTHMAX
COMMON /IBL12/ KMAX1,KMAX2,NCONV
COMMON /BL4/ P(4,4,200),X(4,200),ZF1M(4,4,10),ZF2M(4,4,10),
1 ZF3M(4,4,10),ZF4M(4,4,10)
COMMON /BL5/ TT(10),MT(10),DT(10),DMT(10)
COMMON /BL6/ Z(4,220),SOE,OSE,ALOAD
COMMON /BL7/ D1,S1
COMMON /BL8/ R(200),GAM(200),DMT(200)
COMMON /BL10/ PHIX(10),PHIT(10),PHI(10)

```

```

00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230

```



```

COMMON /BL11/ QMX1(200),PHEE,T0,T2 00000240
COMMON /BL12/ TDL1,TDEL 00000250
COMMON /BL14/ LAM2, LSD18, LSD1N 00000260
COMMON /BL15/ NU,U1(10),V1(10),W1(10),V2(10),U2(10),W2(10),U3(10), 00000270
1 V3(10),W3(10) 00000280
COMMON /BL17/ DEL 00000290
COMMON /BL19/ TH(36) 00000300
COMMON /BL20/ DEOMX(200) 00000310
COMMON /BL27/ BX3(10),BT3(10),BXT3(10),BE3(10) 00000320
COMMON /BL31/ DELSQ,EXT1(10) 00000330
COMMON /BL32/ TKN,ELAST,CHAR,SIGO 00000340
COMMON /BL100/ TEEQ,$DYNMC 00000350
COMMON /BL101/ ZO(4,220),Z2(4,220),Z3(4,220),DELSO 00000360
COMMON /BL102/ DELOAD 00000370
COMMON /BL103/ MASS(200) 00000380
COMMON /BL110/ TX(10),TTH(10),MX(10),MTH(10),MXT(10), 00000390
1 QS(10) 00000400
COMMON /BL111/ ABZ,ABZO,ABZN,ABZ3,DD2 00000410
***** 00000420
CALL BDB(K,BS,DB,DS,DD) 00000430
IF(K.EQ.KMAX) GO TO 301 00000440
DO 202 MN=1,MNMAXO 00000450
U1(MN)=U2(MN) 00000460
V1(MN)=V2(MN) 00000470
W1(MN)=W2(MN) 00000480
I3=3+(MN-1)*KMAX2 00000490
I2=I3-1 00000500
U2(MN)=Z(1,I3) 00000510
V2(MN)=Z(2,I3) 00000520
W2(MN)=Z(3,I3) 00000530
PHIX(MN)=0.0 00000540
PHIT(MN)=0.0 00000550
PHI(MN)=0.0 00000560
MX(MN)=Z(4,I2)*ABZ3 00000570
MTH(MN)=0.0 00000580
MXT(MN)=0.0 00000590
QS(MN)=0.0 00000600
TX(MN)=0.0 00000610
TTH(MN)=0.0 00000620
TXT(MN)=0.0 00000630
IF(M1.EQ.0) GO TO 203 00000640
CALL INLPOL 00000650
PHIX(M1)=PHEE*ABZO 00000660
PHIT(M1)=-PHEE*ABZO 00000670
QS(M1)=0.0 00000680
NOTE: CARDS SERIALIZED #00000690 THRU #00000800 HAVE BEEN REMOVED. 00000680
IF(MO.EQ.0) GO TO 204 00000680
TX(MO)=TO*ABZ 00000680
00000820

```


234	TTH(M0) =T0*ABZ MTH(M0) =MX(M0) IF(M2.EQ.0) GO TO 205 TX(M2) = T2*ABZ TTH(M2) =-T2*ABZ TXT(M2) =-T2*ABZ MTH(M2) =-MX(M2) MXT(M2) =MTH(M2) GO TO 205	00000830 00000840 00000850 00000860 00000870 00000880 00000890 00000900 00000910 00000920 00000930 00000940 00000950 00000960 00000970 00000980 00000990 00001000 00001010 00001020 00001030 00001040 00001050 00001060 00001070 00001080 00001090 00001100 00001110 00001120 00001130 00001140 00001150 00001160 00001170 00001180 00001190 00001200 00001210 00001220 00001230 00001240 00001250 00001260 00001270 00001280 00001290 00001300
203	IF(M0.EQ.0) GO TO 206 I3=3+(M0-1)*KMAX2 I4=I3+1 CALL TLOAD(1) TX(M0)=BS*SI*((2.*Z(1,I3)-.5*Z(1,I4))/DEL+OMXI(1)*Z(3,I3-1))*ABZ -TT(M0)*ABZ*ALOAD 1	
206	TTH(M0)=TX(M0) MTH(M0)=MX(M0) IF(M2.EQ.0) GO TO 205 I3=3+(M2-1)*KMAX2 I4=I3+1 TX(M2)=BS*DI*((2.*Z(1,I3)-.5*Z(1,I4))/DEL TX(M2) = TX(M2)*ABZ TTH(M2) =-TX(M2) TXT(M2) =-TX(M2) MTH(M2) =-MX(M2) MXT(M2) =-MX(M2) RETURN 205	
301	CONTINUE DO 302 MN=1,MNMAX0 U1(MN)=U2(MN) V1(MN)=V2(MN) W1(MN)=W2(MN) PHIX(MN)=0. PHIT(MN)=0. PHI(MN)=0. IK=KMAX1+(MN-1)*KMAX2 MX(MN)=Z(4,IK)*ABZ3 MTH(MN)=0. MXT(MN)=0. QS(MN)=0. TX(MN)=0. TTH(MN)=0. TXT(MN)=0. IF(M1.EQ.0) GO TO 303 CALL FNLPOL PHIX(M1)=PHEE*ABZ0 PHIT(M1)= PHEE*ABZ0 QS(M1)=0.	
302		


```

C      NOTE: CARDS SERIALIZED #00001310 THRU #00001430 HAVE BEEN REMOVED. 0000130A
      IF(MO.EQ.0) GO TO 304
      TX(MO) = TO*ABZ
      TTH(MO) = TO*ABZ
      MTH(MO) = MX(MO)
304   IF(M2.EQ.0) GO TO 305
      TX(M2) = T2*ABZ
      TTH(M2) = -T2*ABZ
      TTX(M2) = T2*ABZ
      MTH(M2) = -MX(M2)
      MXT(M2) = MX(M2)
      GO TO 305
303   IF(MO.EQ.0) GO TO 306
      IKM=KMAX+(MO-1)*KMAX2
      IM1=IKM-1
      CALL TLOAD(KMAX)
      TX(MO)=BS*SI*((-2.*Z(1,IKM)+.5*Z(1,IM1))/DEL+OMX I(KMAX)*Z(3,IKM+1))
      TX(MO)=BS*SI*((-2.*Z(1,IKM)+.5*Z(1,IM1))/DEL+OMX I(KMAX)*Z(3,IKM+1))
      1  TTH(MO) = TX(MO)
      MTH(MO) = MX(MO)
306   IF(M2.EQ.0) GO TO 305
      IKM=KMAX+(M2-1)*KMAX2
      IM1=IKM-1
      TX(M2)=BS*DI*(-2.*Z(1,IKM)+.5*Z(1,IM1))/DEL
      TX(M2)=TX(M2)*ABZ
      TTH(M2) = -TX(M2)
      TTX(M2) = TX(M2)
      MTH(M2) = -MX(M2)
      MXT(M2) = MX(M2)
305   RETURN
      END

```

```

C      SUBROUTINE PLOTIT(X,Y,NN,MODCUR)
C      *****
C      THIS SUBROUTINE AND THE THREE THAT FOLLOW IT COMPRISE THE SELF-
C      CONTAINED PLOTTING CAPABILITY OF THE PROGRAM, SATANS. THEY RECEIVE
C      THE DATA TO BE PLOTTED, ROUND IT, SCALE IT AND DRAW IT ON THE HIGH-
00000010
00000020
00000030
00000040
00000050

```



```

C ***** GO TO 444 00001920
C ***** 00001930
C GRID IS THE MATRIX USED TO PLOT THE POINTS *****
C ***** 00001940
C ***** 00001950
C ***** 00001960
C ***** 00001970
C ***** 00001980
C ***** 00001990
C ***** 00002000
C ***** 00002010
C ***** 00002020
C ***** 00002030
C ***** 00002040
C ***** 00002050
C ***** 00002060
C ***** 00002070
C ***** 00002080
C ***** 00002090
C ***** 00002100
C ***** 00002110
C ***** 00002120
C ***** 00002130
C ***** 00002140
C ***** 00002150
C ***** 00002160
C ***** 00002170
C ***** 00002180
C ***** 00002190
C ***** 00002200
C ***** 00002210
C ***** 00002220
C ***** 00002230
C ***** 00002240
C ***** 00002250
C ***** 00002260
C ***** 00002270
C ***** 00002280
C ***** 00002290
C ***** 00002300
C ***** 00002310
C ***** 00002320
C ***** 00002330
C ***** 00002340
C ***** 00002350
C ***** 00002360
C ***** 00002370
C ***** 00002380
C ***** 00002390

IF(MODCUR.GT.1) GO TO 444
GRID IS THE MATRIX USED TO PLOT THE POINTS
IERR=0
XMAX=RANGE(1)
XMIN=RANGE(2)
YMAX=RANGE(3)
YMIN=RANGE(4)
*****
C ***** 00002000
C ***** 00002010
C ***** 00002020
C ***** 00002030
C ***** 00002040
C ***** 00002050
C ***** 00002060
C ***** 00002070
C ***** 00002080
C ***** 00002090
C ***** 00002100
C ***** 00002110
C ***** 00002120
C ***** 00002130
C ***** 00002140
C ***** 00002150
C ***** 00002160
C ***** 00002170
C ***** 00002180
C ***** 00002190
C ***** 00002200
C ***** 00002210
C ***** 00002220
C ***** 00002230
C ***** 00002240
C ***** 00002250
C ***** 00002260
C ***** 00002270
C ***** 00002280
C ***** 00002290
C ***** 00002300
C ***** 00002310
C ***** 00002320
C ***** 00002330
C ***** 00002340
C ***** 00002350
C ***** 00002360
C ***** 00002370
C ***** 00002380
C ***** 00002390

DO 30 I=1,KDATA,KKZ
IF(X(I).GT.XMAX.OR.X(I).LT.XMIN.OR.Y(I).GT.YMAX.OR.Y(I).LT.YMIN)
1 IERR=IERR+1
IF(X(I).LE.XMAX) GO TO 205
X(I)=XMAX
GO TO 210
205 IF(X(I).GE.XMIN) GO TO 210
X(I)=XMIN
210 IF(Y(I).LE.YMAX) GO TO 215
Y(I)=YMAX
GO TO 30
215 IF(Y(I).GE.YMIN) GO TO 30
Y(I)=YMIN
30 CONTINUE
*****
C ***** 00002190
C ***** 00002200
C ***** 00002210
C ***** 00002220
C ***** 00002230
C ***** 00002240
C ***** 00002250
C ***** 00002260
C ***** 00002270
C ***** 00002280
C ***** 00002290
C ***** 00002300
C ***** 00002310
C ***** 00002320
C ***** 00002330
C ***** 00002340
C ***** 00002350
C ***** 00002360
C ***** 00002370
C ***** 00002380
C ***** 00002390

PLOTING X AND Y AXIS , IF NECESSARY
*****
JERR=0
X RANGE=XMAX-XMIN
Y RANGE=YMAX-YMIN
IF (Y RANGE.NE.0.) GO TO 298
IF(YMIN.EQ.0.) GO TO 889
YMIN=0.
Y RANGE=YMAX
GO TO 299
298 IF (X RANGE.NE.0.) GO TO 299
IF(XMIN.EQ.0.) GO TO 887
XMIN=0.
X RANGE=XMAX
*****
C ***** 00002330
C ***** 00002340
C ***** 00002350
C ***** 00002360
C ***** 00002370
C ***** 00002380
C ***** 00002390

BLANKING OUT MATRIX-(GRID)
*****
299 DO 300 I=1,61
DO 301 JJ=1,81
301 GRID(I,JJ)=BLANK

```



```

300 CONTINUE
   IF(XMAX*XMINGE.0.) GO TO 222
   IYAXIS=80.*(-XMIN)/XRANGE+1.5
   DO 40 I=1,61
   GRID(I,IYAXIS)=DOT
40  IF(YMAX*YMINGE.0.) GO TO 333
222  IYAXIS=60.*YMAX/YRANGE+1.5
   DO 60 I=1,81
60  GRID(IYAXIS,I)=DOT
C*****
C  COMPUTE PROPER SCALE NUMBERS
C*****
333  XINCR=XRANGE/4.
   YINCR=YRANGE/6.
   XSCALE(1)=XMAX
   XSCALE(5)=XMIN
   DO 80 I=2,4
   XSCALE(I)=XSCALE(I-1)-XINCR
80  IF(ABS(XSCALE(I)).LT.1.E-4) XSCALE(I)=0.
   CONTINUE
   YSCALE(1)=YMAX
   YSCALE(7)=YMIN
   DO 81 I=2,6
   YSCALE(I)=YSCALE(I-1)-YINCR
81  IF(ABS(YSCALE(I)).LT.1.E-4) YSCALE(I)=0.
   CONTINUE
   DO 85 I=1,2
   JJ=6-I
   XT=XSCALE(JJ)
   XSCALE(JJ)=XSCALE(I)
85  XSCALE(I)=XT
C*****
C*****
444  IF(MODCUR.LT.2) JSET=0
   IF(JERR.GT.0) GO TO 885
   JSET=JSET+1
   IF(JSET.GT.4) JSET=1
   DO 700 I=1,KDATA,KKZ
   IPTX=60.*(YMAX-Y(I))/YRANGE+1.5
   IPTY=80.*(X(I)-XMIN)/XRANGE+1.5
   IF(IPTX.GT.61.OR.IPTY.GT.81) GO TO 70
   IF(IPTX.LE.0.OR.IPTY.LE.0) GO TO 70
   GRID(IPTX,IPTY) = XCHAR(JSET)
   GO TO 700
70  IERR=IERR+1
700 CONTINUE
C*****
C  OUTPUT SECTION WITH GRAPH
C*****

```



```

C*****IF(MODCUR.EQ.1,OR.MODCUR.EQ.2) RETURN*****00002880
AXR=ABS(XRANGE)*****00002890
AYR=ABS(YRANGE)*****00002900
IF(AXR.LT.1.E+8.AND.AXR.GE..95) GO TO 400*****00002910
WRITE(6,17) XSCALE*****00002920
FORMAT(12X,1PE10.3,4(10X,E10.3)/15X,***,8('+*****'),'+**')*****00002930
GO TO 401*****00002940
WRITE(6,117) XSCALE*****00002950
FORMAT( 8X,F11.2,4( 9X,F11.2)/15X,***,8('+*****'),'+**')*****00002960
DO 101 IK=1,61*****00002970
IF(MOD(IK-1,10).NE.0) GO TO 92*****00002980
IF(AYR.LT.1.E+8.AND.AYR.GE..95) GO TO 404*****00002990
WRITE(6,18) YSCALE(II),(GRID(IK,IX),IX=1,81),YSCALE(II)*****00003000
FORMAT(3X,1PE10.3,2X,1H+,1X,81A1,1X,1H+,2X,E10.3)*****00003010
GO TO 405*****00003020
WRITE(6,118) YSCALE(II),(GRID(IK,IX),IX=1,81),YSCALE(II)*****00003030
FORMAT(2X,F11.2,'+',81A1,'+',F11.2)*****00003040
II=II+1*****00003050
GO TO 101*****00003060
WRITE(6,19) (GRID(IK,IX),IX=1,81)*****00003070
FORMAT(15X,'*',81A1,'*')*****00003080
CONTINUE*****00003090
IF(AXR.LT.1.E+8.AND.AXR.GE..95) GO TO 402*****00003100
WRITE(6,22) XSCALE*****00003110
FORMAT(15X,***,8('+*****'),'+**'/12X,1PE10.3,4(10X,E10.3),//)*****00003120
GO TO 403*****00003130
WRITE(6,217) XSCALE*****00003140
FORMAT(15X,***,8('+*****'),'+**'/ 8X,F11.2,4( 9X,F11.2),//)*****00003150
IF(IERR.GT.0) WRITE(6,20) IERR*****00003160
FORMAT(10X,'NUMBER OF POINTS OUT OF RANGE =' I4)*****00003170
RETURN*****00003180
WRITE(6,888)*****00003190
FORMAT(' ALL Y VALUES=0. CANNOT SETUP PLOT GRID. CHECK MAX & MIN Y*****00003200
1 WHEN MODCUR=0 OR 1.')*****00003210
JERR=10*****00003220
RETURN*****00003230
WRITE(6,886)*****00003240
FORMAT(' ALL X VALUES=0. CANNOT SETUP PLOT GRID. CHECK MAX & MIN X*****00003250
1 WHEN MODCUR=0 OR 1.')*****00003260
JERR=10*****00003270
RETURN*****00003280
WRITE(6,884)*****00003290
FORMAT(' GRID NOT SETUP WHEN MODCUR LAST 0 OR 1. NO PLOT UNTIL GRID*****00003300
1D PROPERLY SETUP,')*****00003310
RETURN*****00003320
END*****00003330
*****00003340
*****00003350

```



```

COMMON /BL5/      TT(10),MT(10),DT(10),DMT(10)
COMMON /BL6/      Z(4,220),SOE,OSE,ALOAD
COMMON /BL7/      D1,S1
COMMON /BL8/      R(200),GAM(200),OMT(200)
COMMON /BL9/      FFS(4,10),ELIS(4),GEES(4,10)
COMMON /BL14/     LAM2,LSD18,LSD1N
COMMON /BL15/     NU,UI(10),VI(10),W1(10),V2(10),W2(10),U3(10),
1 V3(10),W3(10)
COMMON /BL16/     EPS
COMMON /BL18/     ELI(4),ELL(4)
COMMON /BL27/     BX3(10),BT3(10),BXT3(10),BE3(10)
COMMON /BL28/     EXX3(10),ETT3(10),ETX3(10),EXT3(10),ET3(10)
COMMON /BL29/     BX1(10),BT1(10),BXT1(10),BE1(10),BT2(10),
1 BXT2(10),BE2(10)
COMMON /BL30/     EXX1(10),ETT1(10),ETX1(10),ET1(10),EXX2(10),
1 ETT2(10),ETT2(10),EXT2(10),EX2(10),ET2(10)
COMMON /BL31/     DELSQ,EXT1(10)
COMMON /BL100/    TEEC,$DYNMC
COMMON /BL101/    ZO(4,220),Z2(4,220),Z3(4,220),DELSQ
COMMON /BL102/    DELOAD
COMMON /BL103/    MASS(200)
COMMON /BL104/    ZDOT(4,220)
DIMENSION ELLS(4),FLS(4),ZT(4),IPIVOT(4),INDEX(4,2)
1, CLO(4,4),CL1(4,4),CL2(4,4)
2, TZMAX(4,10),ZDD(4)
EQUIVALENCE (CLO(1),ZF1M(1)),(CL1(1),ZF2M(1)),(CL2(1),ZF3M(1))
C*****
DO 201 I=1,4
DO 201 M=1,MNMAX
MJ=1+(M-1)*KMAX2
TZMAX(I,M)=ABS(Z(I,MJ))
DO 201 K=2,KMAX2
KM=K+(M-1)*KMAX2
AZTST=ABS(Z(I,KM))
IF(AZTST.GT.TZMAX(I,M)) TZMAX(I,M)=AZTST
201 CONTINUE
NCONV=1
IF(ITRMAX.EQ.1) GO TO 66
DO 1 M=1,MNMAX0
I=1+(KMAX+2)*(M-1)
U1(M)=Z(1,I)
V1(M)=Z(2,I)
W1(M)=Z(3,I)
I1=I+1
U2(M)=Z(1,I1)
V2(M)=Z(2,I1)
W2(M)=Z(3,I1)
1 IF(IBCINL.LT.0) GO TO 100
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840

```



```

CALL PHIBET(1)
DO 2 M=1,MNMAX
BX1(M)=BX3(M)
BT1(M)=BT3(M)
BXT1(M)=BXT3(M)
BE1(M)=BE3(M)
CALL TEAETA(1)
DO 3 M=1,MNMAX
EXX1(M)=EXX3(M)
ETT1(M)=ETT3(M)
ETX1(M)=ETX3(M)
EXI1(M)=EXI3(M)
ETI1(M)=ETI3(M)
CALL PHIBET(2)
DO 4 M=1,MNMAX
BX2(M)=BX3(M)
BT2(M)=BT3(M)
BXT2(M)=BXT3(M)
BE2(M)=BE3(M)
CALL TEAETA(2)
DO 5 M=1,MNMAX
EXX2(M)=EXX3(M)
ETT2(M)=ETT3(M)
ETX2(M)=ETX3(M)
EXI2(M)=EXI3(M)
ETI2(M)=ETI3(M)
CALL PHIBET(3)
CALL TEAETA(3)
CONTINUE
IF(IBCINL.LT.0) GO TO 20
CALL BDB(I,BI,DB,D,DD)
GAM1=GAM(1)
CALL TLOAD(1)
DO 8 M=1,MNMAX
IF(ITRMAX.EQ.1) GO TO 67
FFS(1,M)=-TI(M)*ALOAD+OSE*(BX1(M)+BE1(M)+NU*(BT1(M)+BE1(M))*B1
FFS(2,M)=OSE*(BI*DI*
FFS(3,M)=LAM2#GAM1*DI*MT(M)*ALOAD-(EXX1(M)+ETX1(M))*SOE
GO TO 8
67 FFS(1,M)=-TT(M)*ALOAD
FFS(2,M)=0.
FFS(3,M)=LAM2#GAM1*DI*MT(M)*ALOAD
8 FFS(4,M)=0.
DO 9 I=1,4
9 ELIS(I)=ALOAD*EL1(I)
CALL FORCE(1)

```

```

00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
00001000
00001010
00001020
00001030
00001040
00001050
00001060
00001070
00001080
00001090
00001100
00001110
00001120
00001130
00001140
00001150
00001160
00001170
00001180
00001190
00001200
00001210
00001220
00001230
00001240
00001250
00001260
00001270
00001280
00001290
00001300
00001310
00001320

```



```

20 CALL FORCE(2)
DO 10 K=3,KLL
  KP=K+1
  IF(ITRMAX.EQ.1) GO TO 10
  CALL UPDATE
  CALL PHIBET(KP)
  CALL TEAETA(KP)
  CALL FORCE(K)
10 IF(ITRMAX.NE.1) CALL UPDATE
  IF(IBCENL.LT.0) GO TO 120
  IF(ITRMAX.EQ.1) GO TO 11
  CALL PHIBET(KMAX)
  CALL TEAETA(KMAX)
  CALL FORCE(KL)
11 CALL FORCE(KMAX)
DO 12 I=1,4
  ELLS(I)=ALOAD*ELL(I)
12 CALL BDB(KMAX,BL,DB,D,DD)
  GAML=GAM(KMAX)
  FLS(4)=0.
  CALL TLOAD(KMAX)
DO 14 M=1,MNMAX
  IF(M.GT.1) ELLS(1)=0.0
  IF(ITRMAX.EQ.1) GO TO 68
  FLS(1)=-TT(M)*ALOAD+OSE*(BX3(M)+BE3(M)+NU*(BT3(M)+BE3(M)))*BL
  FLS(2)=-OSE*(BL*DI*BX3(M)+ET3(M))
  FLS(3)=LAM2*GAML*DI*MT(M)+ETX3(M)*SOE
GO TO 69
68 FLS(1)=-TT(M)*ALOAD
  FLS(2)=0.
  FLS(3)=LAM2*GAML*DI*MT(M)*ALOAD
69 CONTINUE
  IK=KL+KMAX*(M-1)
  IJ=KMAX*M
  L=M*KMAX2
DO 14 I=1,4
  SUMZ=0.
DO 15 J=1,4
  C*****
  C THE FOLLOWING CARD CAUSES BOUNDARY CONS TO EXIST FOR MODE '0', ONLY
  C*****
  IF(M.NE.1) ELLS(J)=0
  15 SUMZ=SUMZ+ZF1M(I,J,M)*ELLS(J)+ZF2M(I,J,M)*X(J,M)+ZF3M(I,J,M)*
    1X(J,IK)+ZF4M(I,J,M)*FLS(J)
  14 Z(I,L)=SUMZ
  150 DO 16 M=1,MNMAX
    DO 16 L=LS,KMAX
      0001330
      0001340
      0001350
      0001360
      0001370
      0001380
      0001390
      0001400
      0001410
      0001420
      0001430
      0001440
      0001450
      0001460
      0001470
      0001480
      0001490
      0001500
      0001510
      0001520
      0001530
      0001540
      0001550
      0001560
      0001570
      0001580
      0001590
      0001600
      0001610
      0001620
      0001630
      0001640
      0001650
      0001660
      0001670
      0001680
      0001690
      0001700
      0001710
      0001720
      0001730
      0001740
      0001750
      0001760
      0001770
      0001780
      0001790
      0001800

```



```

K=KMAX2-L
KPX=K-1
KZ=K+1
IJ=KPX+(M-1)*KMAX
JK=KZ+(M-1)*KMAX2
KK=JK-1
DO 17 I=1,4
SUMZ=0.
DO 18 J=1,4
SUMZ=SUMZ-P(I,J,IJ)*Z(J,JK)
SUMZ=SUMZ+X(I,IJ)
ASUMZ=ABS(SUMZ)
IF(ASUMZ.GT.1.E+15) ITR=ITRMAX
IF(NCONV.NE.1.OR.ASUMZ.LT.1.E-05) GO TO 17
DELZ=ABS(Z(I,KK)-SUMZ)
TEST=EPS*ITZMAX(I,M)
IF(DELZ.GT.ZTEST) NCONV=0
Z(I,KK)=SUMZ
16 CONTINUE
IF(IBCINL.LT.0) GO TO 30
DO 25 M=1,MNMAX
CALL EFG(1,M)
CALL ABC
IJ=2+(M-1)*KMAX2
IJ1=IJ+1
IJ2=IJ-1
DO 21 I=1,4
SUMZ=0.
DO 22 J=1,4
SUMZ=SUMZ-A(I,J)*Z(J,IJ1)-BEE(I,J)*Z(J,IJ)
ZT(I)=SUMZ+GEES(I,M)
CALL MATINV(C,4,ZT,1,DETERM,IPIVOT,INDEX,4,ISCALE)
DO 23 I=1,4
Z(I,IJ2)=ZT(I)
23 CONTINUE
25 RETURN
100 CALL INLPOL
DO 101 M=1,MNMAXD
U1(M)=U2(M)
V1(M)=V2(M)
W1(M)=W2(M)
IJ=3+KMAX2*(M-1)
U2(M)=Z(1,IJ)
V2(M)=Z(2,IJ)
W2(M)=Z(3,IJ)
101 GO TO 102
120 IF(ITRMAX.NE.1) CALL FNLPOL
CALL FORCE(KL)

```



```

IF(M2.EQ.0) GO TO 122
L=KL+(M2-1)*KMAX
LI=KMAX1+(M2-1)*KMAX2
DO 130 I=1,4
SUM=0.
DO 131 J=1,4
SUM=SUM+CL2(I,J)*X(J,L)
131 SUM=SUM+ABS(SUM)
IF(NCONV.NE.1 .OR. ASUMZ.LT.1.E-05) GO TO 130
DELZ=ABS(Z(I,LI)-SUM)
ZTEST=EPS*TZMAX(I,M2)
IF(DE LZ.GT.ZTEST) NCONV=0
Z(I,LI)=SUM
130 IF(M1.EQ.0) GO TO 123
122 L=KL+(M1-1)*KMAX
LI=KMAX1+(M1-1)*KMAX2
DO 132 I=1,4
SUM=0.
DO 133 J=1,4
SUM=SUM+CL1(I,J)*X(J,L)
133 SUM=SUM+ABS(SUM)
IF(NCONV.NE.1 .OR. ASUMZ.LT.1.E-05) GO TO 132
DELZ=ABS(Z(I,LI)-SUM)
ZTEST=EPS*TZMAX(I,M1)
IF(DE LZ.GT.ZTEST) NCONV=0
Z(I,LI)=SUM
132 IF(M0.EQ.0) GO TO 124
123 L=KL+(M0-1)*KMAX
LI=KMAX1+(M0-1)*KMAX2
DO 134 I=1,4
SUM=0.
DO 135 J=1,4
SUM=SUM+CL0(I,J)*X(J,L)
135 SUM=SUM+ABS(SUM)
IF(NCONV.NE.1 .OR. ASUMZ.LT.1.E-06) GO TO 134
DELZ=ABS(Z(I,LI)-SUM)
ZTEST=EPS*TZMAX(I,M0)
IF(DE LZ.GT.ZTEST) NCONV=0
Z(I,LI)=SUM
134 LS=2
124 GO TO 150
END

```

```

00002290
00002300
00002310
00002320
00002330
00002340
00002350
00002360
00002370
00002380
00002390
00002400
00002410
00002420
00002430
00002440
00002450
00002460
00002470
00002480
00002490
00002500
00002510
00002520
00002530
00002540
00002550
00002560
00002570
00002580
00002590
00002600
00002610
00002620
00002630
00002640
00002650
00002660
00002670
00002680
00002690
00002700

```



```

SUBROUTINE PLOT1(I)
C*****
C THIS SUBROUTINE CALLS PLOTTING ROUTINES FOR APPROPRIATE (USER
C SPECIFIED) INPUT QUANTITIES
C*****
IMPLICIT LOGICAL*1 ($)
COMMON /IBL4/ KMAX,KL
COMMON /BLPLOT/
1 2 3 4 5 6
COMMON /BLPLT1/
1 2 3 4 5 6
C*****
NGKMAX=-KMAX
IF (I.GT.1) GO TO 1
IF (IRADII.EQ.0) GO TO 2
WRITE (6,1000)
CALL PLOTIT (XSTATN,XRADII,NGKMAX,0)
2 WRITE (6,1001)
IF (IGAMMA.EQ.0) GO TO 4
WRITE (6,1000)
CALL PLOTIT (XSTATN,YGAMMA,NGKMAX,0)
4 WRITE (6,1002)
IF (IOMEGS.EQ.0) GO TO 5
WRITE (6,1000)
CALL PLOTIT (XSTATN,YOMEGS,NGKMAX,0)
5 WRITE (6,1003)
IF (IOMEGT.EQ.0) GO TO 6
WRITE (6,1000)
CALL PLOTIT (XSTATN,YOMEGT,NGKMAX,0)
6 WRITE (6,1004)
IF (IDEOMS.EQ.0) GO TO 7
WRITE (6,1000)
CALL PLOTIT (XSTATN,YDEOMS,NGKMAX,0)
7 WRITE (6,1005)
IF (IBSTIF.EQ.0) GO TO 8

```



```

WRITE (6,1000)
CALL PLOTT (XSTATN,YBSTIF,NGKMAX,0)
WRITE (6,1006)
8 IF (IDSTIF.EQ.0) GO TO 9
WRITE (6,1000)
CALL PLOTT (XSTATN,YDSTIF,NGKMAX,0)
WRITE (6,1007)
9 IF (IBBSTIF.EQ.0) GO TO 10
WRITE (6,1000)
CALL PLOTT (XSTATN,YBBSTIF,NGKMAX,0)
WRITE (6,1008)
10 IF (IDDSTIF.EQ.0) GO TO 1
WRITE (6,1000)
CALL PLOTT (XSTATN,YDDSTIF,NGKMAX,0)
WRITE (6,1009)
1 IF (IPR.EQ.0) GO TO 11
WRITE (6,1000)
CALL PLOTT (XSTATN,YPR,NGKMAX,0)
WRITE (6,1010)
11 IF (IPS.EQ.0) GO TO 12
WRITE (6,1000)
CALL PLOTT (XSTATN,YPS,NGKMAX,0)
WRITE (6,1011)
12 IF (IPT.EQ.0) GO TO 13
WRITE (6,1000)
CALL PLOTT (XSTATN,YPT,NGKMAX,0)
WRITE (6,1012)
13 IF (ITT.EQ.0) GO TO 14
WRITE (6,1000)
CALL PLOTT (XSTATN,YTT,NGKMAX,0)
WRITE (6,1013)
14 IF (IMT.EQ.0) GO TO 15
WRITE (6,1000)
CALL PLOTT (XSTATN,YMT,NGKMAX,0)
WRITE (6,1014)
15 IF (IDTT.EQ.0) GO TO 16
WRITE (6,1000)
CALL PLOTT (XSTATN,YDTT,NGKMAX,0)
WRITE (6,1015)
16 IF (IDMT.EQ.0) GO TO 17
WRITE (6,1000)
CALL PLOTT (XSTATN,YDMT,NGKMAX,0)
WRITE (6,1016)
17 RETURN
C*****
1000 FORMAT ('1')
1001 FORMAT ('0',T10,'RADIUS VS STATION')

```

```

00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900

```



```

1002 FORMAT ('0',T10,'GAMMA VS STATION',)
1003 FORMAT ('0',T10,'OMEGA-S VS STATION',)
1004 FORMAT ('0',T10,'OMEGA-THETA VS STATION',)
1005 FORMAT ('0',T10,'DEOMEGA-S VS STATION',)
1006 FORMAT ('0',T10,'B-STIFFNESS VS STATION',)
1007 FORMAT ('0',T10,'D-STIFFNESS VS STATION',)
1008 FORMAT ('0',T10,'DB-STIFFNESS VS STATION',)
1009 FORMAT ('0',T10,'DD-STIFFNESS VS STATION',)
1010 FORMAT ('0',T10,'NORMAL LOADING VS STATION',)
1011 FORMAT ('0',T10,'MERIDIO-TANGENTIAL LOADING VS STATION',)
1012 FORMAT ('0',T10,'CIRCUMFERENTIAL LOADING VS STATION',)
1013 FORMAT ('0',T10,'THERMAL LOADING VS STATION',)
1014 FORMAT ('0',T10,'THERMAL BENDING VS STATION',)
1015 FORMAT ('0',T10,'DE-THERMAL BENDING VS STATION',)
1016 FORMAT ('0',T10,'DE-THERMAL-BENDING VS STATION',)
C*****
END
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
00001000
00001010
00001020
00001030
00001040
00001050
00001060
00001070

```

```

C*****
SUBROUTINE PMATRIX
C*****
THIS SUBROUTINE CALLS THE SUBROUTINES HJ(K,MN), EFG(K,MN), ABC,
AND PANDD(K,MN) TO SET UP THE P, P-BAR AND P-HAT MATRICES GIVEN
BY EQUATIONS (30) OF REFERENCE (3).
INTERNALLY, MATRICES DL, DG AND DF ARE SET UP FOR THE CALCULA-
TION OF X(1) GIVEN BY EQUATION (31A) OF REF. (3), WHERE

$$X(1) = DL*SMALL-L(1) + DG*SMALL-G(1) + DF*SMALL-F(1)$$

THE SPECIAL P MATRIX FOR A SHELL WITH AN INITIAL POLE IS ALSO
COMPUTED HERE.
MATRICES ZF1M,ZF2M,ZF3M,ZF4M ARE SET UP FOR THE CALCULATION OF
Z(K+1) GIVEN BY EQUATION (31B) OF REF. (3), WHERE

$$Z(K+1)=ZF1M*SMALL-L(K) + ZF2M*X(K) + ZF3M*X(K-1) + ZF4M*SMALL-F(K)$$

IF THE SHELL HAS A FINAL POLE, THE MATRICES CLO,CL1,CL2 ARE
PREPARED FOR THE CALCULATION OF Z(K)
C*****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190

```



```

DO 6 L=1,4
SUMCB=SUMOB+POTA(I,L)*P8TA(L,J)
SUMOA=SUMOA+POTA(I,L)*PATA(L,J)
SUMOC=SUMOC+POTA(I,L)*C(L,J)
6 DLL(I,J)=SUMOB+POTA(I,J)+CAPLI(I,J)
PTR(I,J)=SUMOA
5 DGG(I,J)=SUMOC
CALL MATINV(DLL,4,PTR,4,DETERM,IPIVOT,INDEX,4,ISCALE)
DO 1 I=1,4
DO 1 J=1,4
SUMD=0.
SUME=0.
DO 7 L=1,4
SUMD=SUMD+DLL(I,L)*DGG(L,J)
7 SUME=SUME-DLL(I,L)*OMEG1(L,J)
DL(I,J,MN)=DLL(I,J)
DG(I,J,MN)=SUMD
DF(I,J,MN)=SUME
IJ=1+KMAX*(MN-1)
1 P(I,J,IJ)=PTR(I,J)
GO TO 20
10 NN=IABS(N(MN))
IJ=1+KMAX*(MN-1)
DO 14 I=1,4
X(I,IJ)=0.
DO 14 J=1,4
P(I,J,IJ)=0.
GO TO 11
IF(NN.GT.3) GO TO 11
IF(NN.GT.2) GO TO 90
IF(NN.GT.1) GO TO 12
IF(NN.GT.0) GO TO 13
P(3,3,IJ)=-1.
P(4,4,IJ)=-1.
M2=MN
GO TO 11
12 P(4,4,IJ)=-1.
GO TO 11
90 M3=MN
GO TO 11
13 P(1,1,IJ)=-1.
P(2,1,IJ)=1.
IF(N(MN).LT.0) P(2,1,IJ)=-1.
M1=MN
11 CONTINUE
20 KLAST=KMAX
IF(IBCFLN.LT.0) KLAST=KL

```



```

DO 23 K=2,KLAST
DO 23 MN=MNINIT,MNMAX
CALL EFG(K,MN)
CALL ABC
CALL PANDD(K,MN)
IF(IBCFL.LT.O) GO TO 30
DO 40 MN=MNINIT,MNMAX
IKL=MN*KMAX-1
CALL HJ(KMAX,MN)
DO 41 I=1,4
DO 41 J=1,4
SUMO=0.
SUMP=0.
SUMJ=0.
DO 42 L=1,4
SUMO=SUMO+OMEGL(I,L)*H(L,J)
SUMP=SUMP+P(I,L,IKL)*P(L,J,JKL)
SUMJ=SUMJ+OMEGL(I,L)*JAY(L,J)
PATA(I,J)=SUMO
PBTA(I,J)=UNIT(I,J)-SUMP
PJTA(I,J)=SUMJ+CAPLL(I,J)
DO 43 I=1,4
DO 43 J=1,4
SUMOP=0.
SUMJP=0.
SUMCM=0.
DO 44 L=1,4
SUMOP=SUMOP+PATA(I,L)*PBTA(L,J)
SUMJP=SUMJP+PJTA(I,L)*P(L,J,JKL)
SUMCM=SUMCM-PATA(I,L)*P(L,J,IKL)
ZF1(I,J)=SUMOP-SUMJP
ZF2(I,J)=SUMCM-PJTA(I,J)
CALL MATINV(ZF1,4,ZF2,4,DETERM,PIVOT,INDEX,4,ISCALE)
DO 45 I=1,4
DO 45 J=1,4
SZF3=0.
SZF4=0.
DO 46 L=1,4
SZF3=SZF3+ZF1(I,L)*PATA(L,J)
SZF4=SZF4-ZF1(I,L)*OMEGL(L,J)
ZF3M(I,J,MN)=SZF3
ZF4M(I,J,MN)=SZF4
ZF1M(I,J,MN)=ZF1(I,J)
ZF2M(I,J,MN)=ZF2(I,J)
CONTINUE
RETURN
DO 31 MN=MNINIT,MNMAX

```



```

IKL=MN*KMAX-1
NN=IABS(N(MN))
IF(NN.GT.3) GO TO 31
IF(NN.GT.2) GO TO 300
IF(NN.GT.1) GO TO 33
IF(NN.GT.0) GO TO 34
M0=MN
DO 35 J=1,4
DO 35 I=1,4
CLO(I,J)=0.
ZFPO(I,J)=0.
ZFPO(1,1)=1.
ZFPO(2,2)=1.
ZFPO(3,1)=P(3,1,IKL)
ZFPO(3,2)=P(3,2,IKL)
ZFPO(3,3)=P(3,3,IKL)+1.
ZFPO(3,4)=P(3,4,IKL)
ZFPO(4,1)=P(4,1,IKL)
ZFPO(4,2)=P(4,2,IKL)
ZFPO(4,3)=P(4,3,IKL)
ZFPO(4,4)=P(4,4,IKL)+1.
CLO(3,3)=1.
CLO(4,4)=1.
CALL MATINV(ZFPO,4,CLO,4,DETERM,IPIVOT,INDEX,4,ISCALE)
GO TO 31
300 M3=MN
GO TO 31
34 M1=MN
DO 60 J=1,4
DO 60 I=1,4
CLO(I,J)=0.
ZFPI(I,J)=0.
ZFPI(1,1)=P(1,1,IKL)+1.
ZFPI(1,2)=P(1,2,IKL)
ZFPI(1,3)=P(1,3,IKL)
ZFPI(1,4)=P(1,4,IKL)
ZFPI(2,1)=1.
ZFPI(2,2)=-1.
ZFPI(2,3)=1.
ZFPI(3,3)=1.
ZFPI(4,4)=1.
CLO(1,1)=1.
CALL MATINV(ZFPI,4,CLO,4,DETERM,IPIVOT,INDEX,4,ISCALE)
GO TO 31
33 M2=MN
DO 70 J=1,4
DO 70 I=1,4
CLO(I,J)=0.

```

```

00001630
00001640
00001650
00001660
00001670
00001680
00001690
00001700
00001710
00001720
00001730
00001740
00001750
00001760
00001770
00001780
00001790
00001800
00001810
00001820
00001830
00001840
00001850
00001860
00001870
00001880
00001890
00001900
00001910
00001920
00001930
00001940
00001950
00001960
00001970
00001980
00001990
00002000
00002005
00002010
00002020
00002030
00002040
00002050
00002060
00002070
00002080
00002090

```



```

70 ZFP2(I,J)=0.
   ZFP2(1,1)=1.
   ZFP2(2,2)=1.
   ZFP2(3,3)=1.
   ZFP2(4,1)=P(4,1,IKL)
   ZFP2(4,2)=P(4,2,IKL)
   ZFP2(4,3)=P(4,3,IKL)
   ZFP2(4,4)=P(4,4,IKL)+1.
   CL2(4,4)=1.
   CALL MATINV(ZFP2,4,CL2,4,DETERM,IPIVOT,INDEX,4,ISCALE)
31 CONTINUE
   RETURN
   END
00002100
00002110
00002120
00002130
00002140
00002150
00002160
00002170
00002180
00002190
00002200
00002210
00002220

```

```

C***** SUBROUTINE UPDATE *****
C      THIS SUBROUTINE UPDATES THE STORAGE LOCATIONS OF THE BETA'S AND
C      ETA'S. IT IS CALLED IN SUBROUTINE XANDZ AFTER A MERIDIAN
C      STATION CHANGE. *****
C***** CCOMMON /IBL1/ MNMAX *****
COMMON /BL27/ BX3(10),BT3(10),BXI3(10),BE3(10)
COMMON /BL28/ EXX3(10),ETT3(10),ETX3(10),EX3(10),ET3(10)
COMMON /BL29/ BX1(10),BT1(10),BXI1(10),BE1(10),BX2(10),BT2(10),
1 BXI2(10),BE2(10)
1 COMMON /BL30/ EXX1(10),ETT1(10),ETX1(10),EX1(10),ET1(10),EXX2(10),
ETT2(10),ETX2(10),EX2(10),ET2(10)
C***** CCOMMON /BL31/ *****
COMMON /BL31/ DELSQ,EXI1(10)
DO 1 M=1,MNMAX
  BX1(M)=BX2(M)
  BT1(M)=BT2(M)
  BXI1(M)=BXI2(M)
  BE1(M)=BE2(M)
  BX2(M)=BX3(M)
  BT2(M)=BT3(M)
  BXI2(M)=BXI3(M)

```



```

00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380

```

```

BE2(M)=BE3(M)
EXX1(M)=EXX2(M)
ETT1(M)=ETT2(M)
ETX1(M)=ETX2(M)
EX1(M)=EX2(M)
ET1(M)=ET2(M)
EXX2(M)=EXX3(M)
ETT2(M)=ETT3(M)
ETX2(M)=ETX3(M)
EX12(M)=EX13(M)
EX2(M)=EX3(M)
1 RETURN
END

```

```

SUBROUTINE INLPOL
C *****
C THIS SUBROUTINE COMPUTES THE NON-LINEAR TERMS BETA-SUB S,
C -SUB THETA, -SUB S-THETA, AETA-SUB S-S AND -SUB THETA-S AT AN
C INITIAL POLE.
C *****
COMMON /IBL1/ MNMAX
COMMON /IBL3/ M0,M1,M2,M3
COMMON /IBL12/ KMAX1,KMAX2,NCONV
COMMON /IBL13/ ITRMAX,LSMAX
COMMON /BL5/ T(10),EMT(10),DT(10),DMT(10)
COMMON /BL6/ Z(4,220),SOE,OSE,ALOAD
COMMON /BL7/ D1,S1
COMMON /BL11/ OMXI(200),PHEE,T0,T2
COMMON /BL17/ DEL
COMMON /BL29/ BXT1(10),BXT1(10),BE1(10),BX2(10),BT2(10),
1 BXT2(10),BE2(10)
COMMON /BL30/ EXX1(10),ETT1(10),ETX1(10),EX1(10),ETT(10),EXX2(10),
1 ETT2(10),ETX2(10),EX2(10),ET2(10)
COMMON /BL31/ DELSQ,EXT1(10)
C *****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210

```



```

DO 1 MN=1, MNMAX
BX1 (MN)=0.
BT1 (MN)=0.
BXT1 (MN)=0.
BE1 (MN)=0.
EX1 (MN)=0.
ET1 (MN)=0.
ETX1 (MN)=0.
EXX1 (MN)=0.
RETURN
1 IF (M1.EQ.0) RETURN
I2=2+(M1-1)*KMAX2
I3=I2+1
I4=I3+1
PHEE=((1.5*Z(3,I2)-2.*Z(3,I3)+.5*Z(3,I4))/DEL+OMXI(1)*Z(1,I2)
BET=.5*PHEE**2
IF (ITRMAX.EQ.1) BET=0.
T2=0.
IF (M2.EQ.0) GO TO 2
CALL BDB(1,B,DB,D,DD)
I2=2+(M2-1)*KMAX2
I3=I2+1
I4=I3+1
T2=B*DI*((-1.5*Z(1,I2)+2.*Z(1,I3)-.5*Z(1,I4))/DEL+.5*SOE*BET)
Q1=-.5*PHEE*T2
BX1 (M2)=BET
BT1 (M2)=-BET
BXT1 (M2)=-BET
ETX1 (M1)=Q1
IF (M3.EQ.0) GO TO 2
EXX1 (M3)=Q1
ETX1 (M3)=-Q1
T0=0.
2 IF (M0.EQ.0) GO TO 3
BX1 (M0)=BET
BT1 (M0)=BET
CALL BDB(1,B,DB,D,DD)
CALL TLOAD(1)
I2=2+(M0-1)*KMAX2
I3=I2+1
I4=I3+1
T0=B*SI*((-1.5*Z(1,I2)+2.*Z(1,I3)-.5*Z(1,I4))/DEL+OMXI(1)*Z(3,I2)
1+.5*SOE*BET)-TT(M0)*ALOAD
3 EXX1 (M1)=PHEE*(T0+.5*T2)
END

```



```

SUBROUTINE FNLPOL
C*****
C      THIS SUBROUTINE COMPUTES THE NON-LINEAR TERMS BETA-S,
C      -SUB THETA, -SUB S-THETA, AETA-SUB S-S, AND -SUB THETA-S AT A
C      FINAL POLE.
C*****
COMMON /IBL1/ MNMAX
COMMON /IBL3/ MO,M1,M2,M3
COMMON /IBL4/ KMAX,KL
COMMON /IBL12/ KMAX1,KMAX2,NCONV
COMMON /IBL13/ ITRMAX,LSMAX
COMMON /BL5/ TT(10),EMT(10),DT(10),DMT(10)
COMMON /BL6/ Z(4,220),SOE,OSE,ALOAD
COMMON /BL7/ D1,S1
COMMON /BL11/ OMXI(200),PHEE,TO,T2
COMMON /BL17/ DEL
COMMON /BL27/ BX3(10),BT3(10),BXI3(10),BE3(10)
COMMON /BL28/ EXX3(10),ETI3(10),ETX3(10),ETI3(10),EXI3(10),EI3(10)
C*****
DO 1 MN=1,MNMAX
  BX3(MN)=0.
  BT3(MN)=0.
  BXI3(MN)=0.
  BE3(MN)=0.
  EX3(MN)=0.
  ET3(MN)=0.
  ETX3(MN)=0.
  EXX3(MN)=0.
  CALL BDB(KMAX,B,DB,D,DB)
  IF(M1.EQ.0) RETURN
  KM=KMAX1+(M1-1)*KMAX2
  KM1=KM-1
  KM2=KM-2
  PHEE=-(1.5*Z(3,KM)-2.*Z(3,KM1)+.5*Z(3,KM2))/DEL+OMXI(KMAX)*Z(1,KM)
  BET=.5*PHEE**2
  IF(ITRMAX.EQ.1) BET=0.
  T2=0.
  IF(M2.EQ.0) GO TO 2
  IF(M2.EQ.0) KM=KMAX1+(M2-1)*KMAX2
1
C*****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390

```



```

KM1=KM-1
KM2=KM-2
T2=B*D1*((1.5*Z(1,KM)-2.*Z(1,KM1)+.5*Z(1,KM2))/DEL+.5*SOE*BET)
Q1=.5*PHEE*T2
BX3(M2)=BET
BT3(M2)=-BET
BTX3(M2)=BET
ETX3(M1)=Q1
IF(M3.EQ.0) GO TO 2
EXX3(M3)=Q1
ETX3(M3)=-Q1
T2=0
2 IF(MJ.EQ.0) GO TO 3
CALL TLOAD(KMAX)
KM=KMAX1+(M0-1)*KMAX2
KM1=KM-1
KM2=KM-2
BX3(M0)=BET
BT3(M0)=BET
T2=B*S1*((1.5*Z(1,KM)-2.*Z(1,KM1)+.5*Z(1,KM2))/DEL+OMXI(KMAX)*
1 Z(3,KM)+.5*SOE*BET)-TT(M0)*ALOAD
3 EXX3(M1)=PHEE*(T2+.5*T2)
RETURN
END

```

```

00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630

```

```

SUBROUTINE PHIBET(K)
C *****
C THIS SUBROUTINE CALCULATES THE PHI'S AND CARRIES OUT THE
C MULTIPLYING AND SUMMATION PROCEDURE FOR COMPUTING THE BETA
C NON-LINEAR TERMS FOR A GIVEN MERIDIONAL STATION K. THE ARRAYS
C IS, JS, IC, JS, IJS, MAXS, MAXD, MAXSY ARE PREPARED IN SUB-
C ROUTINE MODES AND USED HERE.
C *****
COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(10), MNINIT
COMMON /IBL4/ KMAX, KL
COMMON /IBL7/ MNMAXO, MAXD(10), MAXS(10), MAXSY(10), IS(10,10),
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120

```



```

1 COMMON /IBL12/ JS(10,10),ID(10,10),JD(10,10),IJS(10,
COMMON /IBL13/ KMAX1,KMAX2,NCONV
COMMON /BL6/ ITRMAX,LSMAX
COMMON /BL8/ Z(4,220),SOE,OSE,ALOAD
COMMON /BL10/ R(200),GAM(200),OMT(200)
COMMON /BL11/ PHIX(10),PHIT(10),PHI(10)
COMMON /BL12/ OMT(200),PHEE,T0,T2
COMMON /BL15/ TDLI,TDEL
COMMON /BL27/ NU,U1(10),V1(10),V2(10),W2(10),U3(10),
COMMON /BL27/ BX3(10),BT3(10),EXT3(10),BE3(10)
C*****
OX=OMXI(K)
OT=OMT(K)
RRA=1./R(K)
GA=GAM(K)
KP2=K+2
DO 1 M=1,MNMAXO
EN=N(M)
IK=KP2+(M-1)*KMAX2
U3(M)=Z(1,IK)
V3(M)=Z(2,IK)
W3(M)=Z(3,IK)
PHIX(M)=-TDLI*(W3(M)-W1(M))+OX*U2(M)
PHIT(M)=EN*W2(M)*RRA+V2(M)*OT
PHI(M)=(TDLI*(V3(M)-V1(M))+GA*V2(M)+EN*U2(M)*RRA)*.5
1 IF(ITRMAX.EQ.1) RETURN
DO 9 M=1,MNMAX
SMD=0.
SMT=0.
SMR=0.
SMF=0.
IF(N(M).EQ.0) GO TO 20
MAXL=MAXS(M)
IF(MAXL.EQ.0) GO TO 2
DO 3 L=1,MAXL
I=IS(L,M)
J=JS(L,M)
SMD=SMD+PHIX(I)*PHIX(J)
SMT=SMT-PHIT(I)*PHIT(J)
SMR=SMR+PHIX(I)*PHIT(J)+PHIX(J)*PHIT(I)
SMF=SMF-PHI(I)*PHI(J)
3 MAXL=MAXD(M)
2 IF(MAXL.EQ.0) GO TO 4
DO 5 L=1,MAXL
I=ID(L,M)
J=JD(L,M)
SMD=SMD+PHIX(I)*PHIX(J)

```



```

SMT=SMT+PHIT(I)*PHIT(J)
SMR=SMR-PHIX(I)*PHIT(J)+PHIX(J)*PHIT(I)
SMF=SMF+PHI(I)*PHI(J)
5 IF(MAXSY(M)-EQ.0) GO TO 10
4 I=IJS(M)
SMT=SMO+PHIX(I)**2/2.
SMR=SMT-PHIT(I)**2/2.
SMT=(SMR+PHIX(I)*PHIT(I))
SMF=SMF-PHI(I)**2/2.
GO TO 10
20 DO 21 L=1,MNMAXO
SMO=SMO+PHIX(L)**2
SMT=SMT+PHIT(L)**2
21 SMF=SMF+PHI(L)**2
IF(M.GT.MNMAXO) GO TO 11
SMO=SMO+PHIX(M)**2
11 BX3(M)=SMO*.5
BT3(M)=SMT*.5
BE3(M)=SMF*.5
BXT3(M)=0.
GO TO 9
10 BX3(M)=SMO
BT3(M)=SMT
BXT3(M)=SMR*.5
BE3(M)=SMF
9 CONTINUE
RETURN
END

```

```

00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880

```

```

C** SUBROUTINE EFG(K,MN)
C** THIS SUBROUTINE PREPARES THE ELEMENTS OF THE E, F, AND G
C** FOR EACH MERIDIAN STATION K AND FOR EACH FOURIER MODE, MN.
C** IMPLICIT LOGICAL*1(I)
C** REAL NU,N,LAM2,LSD18,LSD19,MINIT
COMMON /IBL2/NU(10),MINIT
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080

```



```

COMMON /BL8/ R(200),GAM(200),OMT(200)
COMMON /BL11/ OMT(200),PHEE,T0,T2
COMMON /BL14/ LAM2,LSD18,LSD1N
COMMON /BL15/ NU,U1(10),V1(10),W1(10),V2(10),U2(10),W2(10),U3(10),
1 V3(10),W3(10)
COMMON /BL20/ DEOMX(200)
COMMON /BL25/ E(4,4),F(4,4),G(4,4)
COMMON /BL100/ TEO,$DYNMC
COMMON /BL101/ ZO(4,220),Z2(4,220),Z3(4,220),DELS
COMMON /BL102/ DELOAD
COMMON /BL103/ MASS(200)
C**
N=NN(MN)
MASS=MASS(K)
CALL BDB(K,B,DB,D,DD)
E(1,1)=B
E(1,2)=0.
E(1,3)=0.
E(1,4)=0.
E(2,1)=0.
DI=(1.-NU)
RA=R(K)
GA=GAM(K)
OX=OMX(K)
OT=OMT(K)
DEX=DEOMX(K)
REX={3.*OT-OX}
GA2=GA**2
RXE={3.*OX-OT}
OTX=OT*OX
DNLR=LAM2*D*N*D1/(2.*RA)
DDNLR=DNLR*DD/D
E(2,2)=B*D1/2.+LAM2*D*D1*REX**2/8.
E(2,3)=DNLR*REX
E(2,4)=J.
E(3,1)=0.
E(3,2)=E(2,3)
RAN=(N/RA)**2
E(3,3)=LAM2*D*D1*(2.*RAN+(1.+NU)*GA2)
E(3,4)=LAM2
E(4,1)=0.
E(4,2)=0.
E(4,3)=-D
E(4,4)=0.
F(1,1)=GA*B+DB
F(1,2)=(1.+NU)*B*N/(2.*RA)+DNLR*REX*RXE/4.
F(1,3)=B*(OX+NU*OT)+LAM2*D*D1*((1.+NU)*GA2*OX+RAN*RXE/2.)
F(1,4)=LAM2*OX
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560

```



```

F(2,1)=-F(1,2)
F(2,2)=(D1/2.)*(GA*B+DB)-(LAM2*D*D1*REX/8.)*(2.*DEX-GA*(5.*OX
1-3.*OT))+LAM2*DD*D1*REX**2/8.
F(2,3)=DNLR*(2.*(1.+NU)*GA*OT-DEX+3.*GA*(OX-OT))+DDNLR*REX
F(2,4)=0.
F(3,1)=-F(1,3)
F(3,2)=DNLR*(3.*GA*OX-GA*OT*(5.+2.*NU)-DEX)+DDNLR*REX
F(3,3)=-LAM2*D*D1*((1.+NU)*(2.*GA*OX*OT+GA**3)+2.*GA*GAN)
1+LAM2*DD*D1*((1.+NU)*GA2+2.*GAN)
F(3,4)=LAM2*GA*(2.-NU)
F(4,1)=D*OX
F(4,2)=0.
F(4,3)=-D*NU*GA
F(4,4)=0.
G(1,1)=NU*DB*GA-NU*B*OTX-B*GA2-D1*B*GAN/2.-LAM2*D*D1*((1.+NU)*GA2*
1OX**2+RXE**2*GAN/8.)
2-2.*MAS/DELS
G(1,2)=NU*DB/RA-(3.-NU)/(2.*RA)*GA*B*N-DNLR*2.*GA*(REX*RXE/8.
1+(1.+NU)*OTX)
G(1,3)=B*(DEX+GA*(OX-OT))+DB*(OX+NU*OT)-LAM2*D*D1*GA*GAN*(RXE/2.+(
11.+NU)*OX)
G(1,4)=LAM2*D1*GA*OX
G(2,1)=-B*GAN*(3.-NU)/(2.*RA)-D1*N*DB/(2.*RA)+DNLR*2.*(-1.*(1.+
1NU)*GA*OTX+GA/8.*(6.*OTX-7.*OX**2-3.*OT**2)-DEX/4.*(5.*OT-3.*OX))
2-DDNLR/4.*REX*RXE
G(2,2)=-GA*F(2,2)+D1/2.*B*OTX-B*GAN-LAM2*D*D1*((1.+NU)*OT**2*GAN
1-OTX/8.*REX**2)
2-2.*MAS/DELS
G(2,3)=-B*N*(OT+NU*OX)/RA+DNLR*(GA*DEX-2.*GA2*OX-2.*(1.+NU)*OT
1*GAN+REX*(GA2+OTX))-DDNLR*REX*GA
G(2,4)=-NU*LAM2*OT*N/RA
G(3,1)=-B*GA*(OT+NU*OX)+LAM2*D*D1*(GA*(1.+NU)*(-GA*DEX+GA2*OX
1-OX*GAN+2.*OTX*OX)+GAN/2.*(GA*OX-GA*OT-3.*DEX))
2-LAM2*DD*D1*((1.+NU)*GA2*OX+GAN/2.*RXE)
G(3,2)=-B*N*(OT+NU*OX)/RA+DNLR*(2.*(1.+NU)*(OTX*OT-GA2*OX+2.*GA2
1*OT-OT*GAN)+GA*DEX+3.*GA2*(OT-OX)+OTX*REX)-DDNLR*(2.*(1.+NU)*GA
2*OT+GA*REX)
G(3,3)=-B*(OX**2+2.*NU*OTX+OT**2)+LAM2*D*D1*GAN*((1.+NU)*(OTX-RAN
1+2.*GA2)+2.*(GA2+OTX))-LAM2*DD*D1*GAN*(3.+NU)*GA
2-2.*MAS/DELS
G(3,4)=-LAM2*(D1*OTX+NU*GAN)
G(4,1)=D*(DEX+NU*GA*OX)
G(4,2)=D*NU*N*OT/RA
G(4,3)=D*NU*GAN
G(4,4)=-1.
RETURN
END

```



```

C***** SUBROUTINE TEAETA(K)
C***** THIS SUBROUTINE CALCULATES THE INPLANE FORCES AND CARRIES OUT
C***** THE MULTIPLYING AND SUMMATION PROCEDURE FOR COMPUTING THE AETA
C***** NON-LINEAR TERMS FOR A GIVEN MERIDIONAL STATION K. THE ARRAYS
C***** IS, JS, ID, JS, IJS, MAXS, MAXD, MAXSY PREPARED IN SUBROUTINE
C***** MODES ARE USED HERE
C*****
C***** REAL NU, MT
COMMON /IBL1/ MNMAX
COMMON /IBL2/ MN(10), MNINIT
COMMON /IBL7/ MNMAXO, MAXD(10), MAXS(10), MAXSY(10), IS(10,10),
1 JS(10,10), ID(10,10), JD(10,10), IJS(10,10)
COMMON /IBL8/ LSTEP, ITR
COMMON /IBL13/ ITRMAX, L, SMAX
COMMON /IBL5/ TT(10), MT(10), DT(10), DMT(10)
COMMON /BL6/ Z(4,220), SOE, DSE, ALOAD
COMMON /BL7/ D1, S1
COMMON /BL8/ R(200), GAM(200), OMT(200)
COMMON /BL10/ PHIX(10), PHIT(10), PHI(10)
COMMON /BL11/ QMXI(200), PHEE, TO, T2
COMMON /BL12/ TDLI, TDEL
COMMON /BL15/ NU, UI(10), V1(10), V2(10), U2(10), W2(10), U3(10),
1 V3(10), W3(10)
COMMON /BL27/ BX3(10), BT3(10), BXT3(10), BE3(10)
COMMON /BL28/ EXX3(10), ETX3(10), ETX3(10), ET3(10)
C***** DIMENSION ION TX(10), TTH(10), TXT(10)
C*****
C***** RA=1./R(K)
C***** GA=GAM(K)
C***** GX=OMXI(K)
C***** OT=OMT(K)
C***** CALL BDB(K, BS, DB, DS, DD)
C***** DO I M=1, MNMAXO
C***** EN=N(M)
C***** CALL TLOAD(K)
C***** TTS=TT(M)*ALOAD
C*****
00000010
00000020
00000030
00000033
00000040
00000050
00000055
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370

```



```

EX=(U3(M)-U1(M))*TDLI+OX*W2(M)+OSE*(BX3(M)+BE3(M))
ET= EN *V2(M)*RRA+GA*U2(M)+OT*W2(M)+OSE*(BT3(M)+BE3(M))
EXT=.5*(TDLI*(V3(M)-V1(M))-EN *U2(M)*RRA-GA*V2(M))+OSE*BT3(M)
TX(M)=BS*(EX+NU*ET)-TTS
TTH(M)=BS*(ET+NU*EX)-TTS
1  TXT(M)=BS*D1*EXT
   DO 9 M=1,MNMAX
   SMS=0.
   SMV=0.
   SME=0.
   SMN=0.
   SMT=0.
   IF(N(M).EQ.0) GO TO 20
   MAXL=MAXS(M)
   IF(MAXL.EQ.0) GO TO 2
   DO 3 L=1,MAXL
   I=IS(L,M)
   J=JS(L,M)
   SMF=SMF+TX(I)*PHIX(J)+TX(J)*PHIX(I)
   SMS=SMS+TTH(I)*PHIT(J)+TTH(J)*PHIT(I)
   SMV=SMV-PHIT(I)*TXT(J)-PHIT(J)*TXT(I)
   SME=SME+PHIX(I)*TXT(J)+PHIX(J)*TXT(I)
   SMN=SMN+TX(I)*PHI(J)+TX(J)*PHI(I)
   SMT=SMT+TTH(I)*PHI(J)+TTH(J)*PHI(I)
   MAXL=MAXD(M)
2  IF(MAXL.EQ.0) GO TO 4
   DO 5 L=1,MAXL
   I=ID(L,M)
   J=JD(L,M)
   SMF=SMF+TX(I)*PHIX(J)+TX(J)*PHIX(I)
   SMS=SMS-TTH(I)*PHIT(J)+TTH(J)*PHIT(I)
   SMV=SMV+PHIT(I)*TXT(J)+PHIT(J)*TXT(I)
   SME=SME-PHIX(I)*TXT(J)+PHIX(J)*TXT(I)
   SMN=SMN-TX(I)*PHI(J)+TX(J)*PHI(I)
   SMT=SMT-TTH(I)*PHI(J)+TTH(J)*PHI(I)
   IF(MAXSY(M).EQ.0) GO TO 10
5 4 I=IJS(M)
   SMF=SMF+TX(I)*PHIX(I)
   SMS=SMS+TTH(I)*PHIT(I)
   SMV=SMV-PHIT(I)*TXT(I)
   SME=SME+PHIX(I)*TXT(I)
   SMN=SMN+TX(I)*PHI(I)
   SMT=SMT+TTH(I)*PHI(I)
   GO TO 10
20 DO 21 L=1,MNMAX
   SMF=SMF+TX(L)*PHIX(L)
21 SMV=SMV+PHIT(L)*TXT(L)

```



```

IF(M.GT.MNMAXD) GO TO 10
SMF=SMF+TX(M)*PHIX(M)
10 EXX3(M)=SMF*.5
   ETT3(M)=SMF*.5
   ETX3(M)=SMV*.5
   EXT3(M)=SME*.5
   EX3(M)=SMN*.5
   ET3(M)=SMT*.5
9  CONTINUE
   DO 30 M=1,MNMAXD
   U1(M)=U2(M)
   V1(M)=V2(M)
   W1(M)=W2(M)
   U2(M)=U3(M)
   V2(M)=V3(M)
   W2(M)=W3(M)
30 RETURN
END

```

```

00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
00001000
00001010
00001020
00001030

```


APPENDIX F

LISTING OF SATANS-II MODIFIED SUBROUTINES AND COMMON STATEMENTS

AS INDICATED IN THE BODY OF THIS REPORT, REPLACE THE FOLLOWING COMMON STATEMENTS IN ALL SUBROUTINES WHERE THEY OCCUR.

```

COMMON /IBL2/ N(25),MN,INIT
COMMON /IBL7/ MNMAXO,MAXD(25),MAXS(25),MAXSY(25),IS(25,25),
1 JS(25,25),ID(25,25),JD(25,25),IJS(25)
COMMON /IBL11/ ICORFL,IPASS,JUMP,MPERFS
COMMON /BL3/ PR(25),PX(25),PT(25)
COMMON /BL4/ P(4,4,951),X(4,951),ZF1M(4,4,25),ZF2M(4,4,25),
1 ZF3M(4,4,25),ZF4M(4,4,25)
COMMON /BL5/ TT(25),MT(25),DT(25),DMT(25)
COMMON /BL6/ Z(4,1001),SOE,OSE,ALOAD
COMMON /BL8/ R(100),GAM(100),OMT(100)
COMMON /BL9/ FFS(4,25),ELIS(4),GEES(4,25)
COMMON /BL10/ PHIX(25),PHIT(25),PHI(25)
COMMON /BL11/ OMXI(100),PHEE,T0,T2
COMMON /BL15/ NU,U1(25),V1(25),W1(25),V2(25),U2(25),W2(25),U3(25),
1 V3(25),W3(25)
COMMON /BL20/ DEOMX(100)
COMMON /BL24/ DL(4,4,25),DG(4,4,25),DF(4,4,25)
COMMON /BL27/ BX3(25),BT3(25),BXT3(25),BE3(25)
COMMON /BL28/ EXX3(25),ETT3(25),ETX3(25),EXT3(25),ET3(25)
COMMON /BL29/ BX1(25),BT1(25),BXT1(25),BE1(25),BT2(25),
1 BXT2(25),BE2(25)
COMMON /BL30/ EXX1(25),ETT1(25),ETX1(25),EXT1(25),EXX2(25),
1 ETT2(25),ETX2(25),EXT2(25),ET2(25)
COMMON /BL31/ DELSQ,EXT1(25)
COMMON /BL34/ DEE(4,4,951),DST(4,4,951)
COMMON /BL101/ ZO(4,1001),Z2(4,1001),Z3(4,1001),DELSO
COMMON /BL103/ MASS(100)
COMMON /BL104/ ZDOT(4,1001)
COMMON /BL110/ TX(25),TTH(25),MX(25),MTH(25),MXT(25),
1 QS(25)

```


AS INDICATED IN THE BODY OF THIS REPORT, REPLACE SUBROUTINES TEAETA, PHIBET, MODES, OUTPUT, INLPOL, FNLPOL AND POLE IN APPENDIX E WITH THOSE THAT FOLLOW.

```

SUBROUTINE TEAETA(K)
C*****
C    THIS SUBROUTINE CALCULATES THE INPLANE FORCES AND CARRIES OUT
C    THE MULTIPLYING AND SUMMATION PROCEDURE FOR COMPUTING THE AETA*
C    NON-LINEAR TERMS FOR A GIVEN MERIDIONAL STATION K. THE ARRAYS
C    'IS', 'JS', 'ID', 'JD', 'IJS', 'MAXS', 'MAXD', AND 'MAXSY' PREPARED
C    IN SUBROUTINE MODES ARE USED HERE
C*****
REAL NU, MT
COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(25), MNINIT
COMMON /IBL7/ MNMAXO, MAXD(25), IDS(25, 25),
1      JS(25, 25), ID(25, 25), JD(25, 25), IJS(25)
COMMON /IBL8/ LSTEP, ITR
COMMON /IBL11/ ICORFL, IPASS, JUMP, MPERFS
COMMON /IBL13/ ITRMAX, LSMAX
COMMON /BL5/ TT(25), MT(25), DT(25), DMT(25)
COMMON /BL6/ Z(4, 1001), SOE, OSE, ALOAD
COMMON /BL7/ D1, S1
COMMON /BL8/ R(100), GAM(100), OMT(100)
COMMON /BL10/ PHIX(25), PHIT(25), PHI(25)
COMMON /BL11/ OMXI(100), PHEE, TO, T2
COMMON /BL12/ TDLI, TDEL
COMMON /BL15/ NU, UI(25), U2(25), U3(25), V1(25), V2(25), V3(25), W1(25),
1      W2(25), W3(25)
COMMON /BL27/ BX3(25), BT3(25), BX3(25), BE3(25)
COMMON /BL28/ EXX3(25), ET3(25), ETX3(25), EX3(25), ET3(25)
COMMON /BLPHS/ PHX(25), PHT(25)
C*****
DIMENSION TX(25), TTH(25), TXT(25)
RRA=1./R(K)
GA=GAM(K)
OX=OMXI(K)
OT=OMT(K)
CALL BDB(K, BS, DB, DS, DD)
DO 1 M=1, MNMAXO
  PHIX(M)=PHIX(M)+PHX(M)
  PHIT(M)=PHIT(M)+PHT(M)
EN=N(M)
CALL TLOAD(K)
0000010
0000020
0000030
0000040
0000050
0000060
0000070
0000080
0000090
0000100
0000110
0000120
0000130
0000140
0000150
0000160
0000170
0000180
0000190
0000200
0000210
0000220
0000230
0000240
0000250
0000260
0000270
0000280
0000290
0000300
0000310
0000320
0000330
0000340
0000350
0000360
0000370
0000380
0000390

```



```

C *** TEST FOR PRESENCE OF SYMMETRIC SUMMATION COMBINATIONS *** 00001360
C *** IF (MAXL.EQ.0) GO TO 2 *** 00001370
C *** IF (MAXL.EQ.0) GO TO 2 *** 00001380
C *** SET UP COUPLING MODES, INDICES AND TEST FOR MODE 1 *** 00001390
C *** SET UP COUPLING MODES, INDICES AND TEST FOR MODE 1 *** 00001400
C *** DO 3 L=1,MAXL *** 00001410
C *** DO 3 L=1,MAXL *** 00001420
C *** I=IS(L,M) *** 00001430
C *** J=JS(L,M) *** 00001440
C *** I=I-1 *** 00001450
C *** JJ=J-1 *** 00001460
C *** IF (I.EQ.1) GO TO 3 *** 00001470
C *** *** 00001480
C *** COMPARE SUMS FOR SYMMETRIC SUM COMBINATION MODES *** 00001490
C *** *** 00001500
C *** SMF=SMF+TX(I)*PHIX(J)+TX(J)*PHIX(I)-PHIX(I)*TX(J) *** 00001510
C *** 1 SMS=SMS+TTH(I)*PHIT(J)+TTH(J)*PHIT(I)+PHIT(I)*TTH(J) *** 00001520
C *** 1 SMV=SMV-PHIT(I)*TX(J)-PHIT(J)*TX(I)+PHIT(I)*TX(J) *** 00001530
C *** 1 SME=SME+PHIX(I)*TX(J)+PHIX(J)*TX(I)+PHIX(I)*TX(J) *** 00001540
C *** 1 SMN=SMN+TX(I)*PHI(J)+TX(J)*PHI(I)+PHI(I)*TX(J) *** 00001550
C *** 1 SMT=SMT+TTH(I)*PHI(J)+TTH(J)*PHI(I)+PHI(I)*TTH(J) *** 00001560
C *** 1 CONTINUE *** 00001570
C *** 3 MAXL=MAXD(M) *** 00001580
C *** TEST FOR PRESENCE OF SYMMETRIC DIFFERENCE COMBINATIONS *** 00001590
C *** IF (MAXL.EQ.0) GO TO 4 *** 00001600
C *** SET UP COUPLING MODES, INDICES AND TEST FOR MODE 1 *** 00001610
C *** SET UP COUPLING MODES, INDICES AND TEST FOR MODE 1 *** 00001620
C *** DO 5 L=1,MAXL *** 00001630
C *** I=ID(L,M) *** 00001640
C *** J=JD(L,M) *** 00001650
C *** I=I-1 *** 00001660
C *** JJ=J-1 *** 00001670
C *** IF (J.EQ.1) GO TO 42 *** 00001680
C *** *** 00001690
C *** COMPARE SUMS FOR SYMMETRIC DIFFERENCE COMBINATIONS *** 00001700
C *** SMF=SMF+TX(I)*PHIX(J)+TX(J)*PHIX(I)+PHIX(I)*TX(J) *** 00001710
C *** 1 SMS=SMS-TTH(I)*PHIT(J)+TTH(J)*PHIT(I)-PHIT(I)*TTH(J) *** 00001720
C *** *** 00001730
C *** *** 00001740
C *** *** 00001750
C *** *** 00001760
C *** *** 00001770
C *** *** 00001780
C *** *** 00001790
C *** *** 00001800
C *** *** 00001810
C *** *** 00001820
C *** *** 00001830

```



```

132 MAXL=MAXD(MP)
IF (MAXL.EQ.0) GO TO 104
C**
C** SET UP COUPLING MODES, INDICES AND TEST FOR MODE 1
C**
DO 105 L=1,MAXL
I=ID(L,MP)
J=JD(L,MP)
II=I-1
JJ=J-1
IF (J.EQ.1) GO TO 123
C**
C** COMPARE SUMS FOR ASYMMETRIC DIFFERENCE COMBINATIONS
C**
SMF=SMF-PHIX(I)*TX(JJ)+PHIX(J)*TX(II)+PHIX(II)*TX(J)
- PHIX(JJ)*TX(I)
1 SMS=SMS+PHIT(I)*TTH(JJ)+PHIT(J)*TTH(II)+PHIT(II)*TTH(J)
+ PHIT(JJ)*TTH(I)
1 SMV=SMV+PHIT(I)*TXT(JJ)+PHIT(J)*TXT(II)+PHIT(II)*TXT(J)
+ PHIT(JJ)*TXT(I)
1 SME=SME+PHIX(I)*TXT(JJ)+PHIX(J)*TXT(II)+PHIX(II)*TXT(J)
+ PHIX(JJ)*TXT(I)
1 SMN=SMN+PHI(I)*TX(JJ)+PHI(J)*TX(II)+PHI(II)*TX(J)
+ PHI(JJ)*TX(I)
1 SMT=SMT+PHI(I)*TTH(JJ)+PHI(J)*TTH(II)+PHI(II)*TTH(J)
+ PHI(JJ)*TTH(I)
GO TO 105
C**
C** EXECUTE BELOW IF J=1 IN DIFF-COMB (OR I=1 IN SUM-COMB)
C**
123 SMF=SMF+(PHIX(I)*TX(II)+PHIX(II)*TX(I))*2.0
SMS=SMS+(PHIT(I)*TTH(II)+PHIT(II)*TTH(I))*2.0
SMV=SMV+(PHIT(I)*TXI(II)+PHIT(II)*TXI(I))*2.0
SME=SME+(PHIX(I)*TXT(II)+PHIX(II)*TXT(I))*2.0
SMN=SMN+(PHI(I)*TX(JJ)+PHI(II)*TX(I))*2.0
SMT=SMT+(PHI(I)*TTH(JJ)+PHI(II)*TTH(I))*2.0
105 CONTINUE
C**
C** TEXT FOR PRESENCE OF SAME-INDEX COMBINATION
C**
104 IF (MAXSY(MP).EQ.0) GO TO 10
C**
C** SET UP COUPLING MODES, INDICES AND COMPILE SUMS
C**
I=IJS(MP)
II=I-1
SMF=SMF+PHIX(I)*TX(II)+PHIX(II)*TX(I)
SMS=SMS-PHIT(I)*TTH(II)+PHIT(II)*TTH(I)
00002800
00002810
00002820
00002830
00002840
00002850
00002860
00002870
00002880
00002890
00002900
00002910
00002920
00002930
00002940
00002950
00002960
00002970
00002980
00002990
00003000
00003010
00003020
00003030
00003040
00003050
00003060
00003070
00003080
00003090
00003100
00003110
00003120
00003130
00003140
00003150
00003160
00003170
00003180
00003190
00003200
00003210
00003220
00003230
00003240
00003250
00003260
00003270

```



```

COMMON /IBL1/ MNMAX
COMMON /IBL2/ N(25),MNINIT
COMMON /IBL4/ KMAX,KL
COMMON /IBL7/ MNMAXO,MAXD(25),MAXS(25),MAXSY(25),IS(25,25),
1 JS(25,25),ID(25,25),JD(25,25),IJS(25)
COMMON /IBL11/ ICORFL,IPASS,JUMP,MPERFS
COMMON /IBL12/ KMAX1,KMAX2,NCONV
COMMON /IBL13/ ITRMAX,LSMAX
COMMON /IBL6/ Z(4,1001),SOE,OSE,ALOAD
COMMON /BL8/ R(100),GAM(100),OMT(100)
COMMON /BL10/ PHIX(25),PHIT(25),PHI(25)
COMMON /BL11/ OMXI(100),PHEE,TO,T2
COMMON /BL12/ TDLI,TDLE
COMMON /BL15/ NU,UI(25),U2(25),U3(25),V1(25),V2(25),V3(25),W1(25),
1 W2(25),W3(25)
COMMON /BL27/ BX3(25),BT3(25),BXT3(25),BE3(25)
COMMON /BLIMP/ PHIXB(951),PHITB(951)
COMMON /BLPHS/ PHX(25),PHT(25)
C*****
OX=OMXI(K)
OT=OMT(K)
RRA=1./R(K)
GA=GAM(K)
KP2=K+2
DO 1 M=1,MNMAXO
EN=N(M)
IK=KP2+(M-1)*MAX2
U3(M)=Z(1,IK)
V3(M)=Z(2,IK)
W3(M)=Z(3,IK)
PHIX(M)=-TDLI*(W3(M)-W1(M))+OX*U2(M)
PHIT(M)=EN*W2(M)*RRA+V2(M)*OT
1 PHIT(M)=(TDLI*(V3(M)-V1(M))+GA*V2(M)+EN*U2(M)*RRA)*.5
IF(I TRMAX.EQ.1) RETURN
IF(JUMP.EQ.2) GO TO 1111
C*****
THIS SECTION HANDLES 'SINGLE-TERM' EXPANSIONS ONLY
C*****
DO 19 M=1,MNMAX
SMO=0.
SMT=0.
SMR=0.
SMF=0.
IF(N(M).EQ.0) GO TO 120
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000261
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550

```



```

C *** DO 60 M=1,MNMAXO 00001040
C *** KP=K+(M-1)*KMAX 00001050
C *** PHX(M)=PHIB(KP) 00001060
C *** PHT(M)=PHTB(KP) 00001070
C *** DO 9 M=1,MNMAX 00001080
C *** SMO=0. 00001090
C *** SMT=0. 00001100
C *** SMR=0. 00001110
C *** SMF=0. 00001120
C *** TEST FOR ASYMMETRIC MODE 00001130
C *** IF (N(M).LT.0) GO TO 101 00001140
C *** 00001150
C *** 00001160
C *** 00001170
C *** 00001180
C *** 00001190
C *** 00001200
C *** 00001210
C *** 00001220
C *** 00001230
C *** 00001240
C *** 00001250
C *** 00001260
C *** 00001270
C *** 00001280
C *** 00001290
C *** 00001300
C *** 00001310
C *** 00001320
C *** 00001330
C *** 00001340
C *** 00001350
C *** 00001360
C *** 00001370
C *** 00001380
C *** 00001390
C *** 00001400
C *** 00001410
C *** 00001420
C *** 00001430
C *** 00001440
C *** 00001450
C *** 00001460
C *** 00001470
C *** 00001480
C *** 00001490
C *** 00001500
C *** 00001510

C *** THIS SECTION HANDLES SYMMETRIC MODE COMBINATIONS ONLY
C ***
C *** TEST FOR ZERO-TH MODE
C *** IF (N(M).EQ.0) GO TO 20
C *** MAXL=MAXS(M)
C *** TEST FOR PRESENCE OF SYMMETRIC SUMMATION COMBINATIONS
C *** IF (MAXL.EQ.0) GO TO 2
C *** SET UP 'COUPLING' MODES, INDICES AND TEST FOR MODE 1
C *** DO 3 L=1,MAXL
C *** I=IS(L,M)
C *** J=JS(L,M)
C *** II=I-I
C *** JJ=J-J
C *** IF (I.EQ.1) GO TO 3
C *** COMPARE SUMS FOR SYMMETRIC SUMMATION COMBINATION MODES
C *** SMO=SMO+PHX(I)*PHX(J)
C *** 1 +PHX(I)*PHX(J)+PHX(J)*PHX(I)
C *** 2 -PHX(II)*(PHX(JJ)+PHX(JJ))-PHX(JJ)*PHX(II)
C *** SMT=SMT-PHT(I)*PHT(J)
C *** 1 -PHT(II)*PHT(JJ)-PHT(JJ)*PHT(II)
C *** 2 +PHT(II)*(PHT(JJ)+PHT(JJ))+PHT(JJ)*PHT(II)
C *** SMR=SMR+PHX(I)*PHX(J)+PHX(J)*PHX(I)+PHX(II)*PHT(J)
C *** 1 +PHX(I)*PHT(JJ)+PHX(JJ)*PHT(II)
C *** 2 +PHX(J)*PHT(I)

```



```

3      +PHIX(II)*(PHI(JJ)+PHT(JJ))+PHIX(JJ)*PHI(II)
4      +PHI(II))+PHX(II)*PHI(JJ)+PHX(JJ)*PHI(II)
1      +PHI(II)*PHI(JJ)
3      CONTINUE
2      MAXL=MAXD(M)
C      *****
C      TEST FOR PRESENCE OF SYMMETRIC DIFFERENCE COMBINATIONS
C      *****
C      IF(MAXL.EQ.0) GO TO 4
C      *****
C      SET UP 'COUPLING' MODES, INDICES AND TEST FOR MODE 1
C      *****
DO 5 L=1,MAXL
I=ID(L,M)
J=JD(L,M)
II=I-1
JJ=J-1
IF(JJ.EQ.1) GO TO 42
C      *****
C      COMPILER SUMS FOR SYMMETRIC DIFFERENCE COMBINATION MODES
C      *****
SMO=SMO+PHIX(I)*PHIX(J)
+PHX(I)*PHIX(II)+PHX(JJ)+PHX(JJ)*PHX(II)
1
2
SMT=SMT+PHI(I)*PHI(J)
+PHI(II)*PHI(JJ)+PHI(JJ)*PHI(II)
1
2
SMR=SMR-PHIX(I)*PHI(II)+PHIX(JJ)*PHI(II)
-PHIX(I)*PHI(JJ)+PHIX(JJ)*PHI(JJ)
+PHIX(J)*PHI(II)
1
2
3
4
SMF=SMF+PHI(I)*PHI(J)
+PHI(II)*PHI(JJ)
1
GO TO 5
C      *****
C      EXECUTE BELOW IF J=1 IN DIFFERENCE COMBINATIONS
C      *****
42
SMO=SMO+(PHIX(1)*PHIX(II)+PHX(1)*PHX(II))*PHI(1)*2.0
SMT=SMT+(PHI(1)*PHI(II)+PHI(II)*PHI(1))*PHI(1)*2.0
SMR=SMR+(PHIX(1)*PHI(II)+PHI(1)*PHI(II))*PHI(1)*2.0
1
SMF=SMF+(PHI(1)*PHI(II))*2.0
5
CONTINUE
C      *****
C      TEST FOR PRESENCE OF SAME-INDEX COMBINATION
C      *****
00001520
00001530
00001540
00001550
00001560
00001570
00001580
00001590
00001600
00001610
00001620
00001630
00001640
00001650
00001660
00001670
00001680
00001690
00001700
00001710
00001720
00001730
00001740
00001750
00001760
00001770
00001780
00001790
00001800
00001810
00001820
00001830
00001840
00001850
00001860
00001870
00001880
00001890
00001900
00001910
00001920
00001930
00001940
00001950
00001960
00001970
00001980
00001990

```



```

4 IF(MAXSY(M).EQ.0) GO TO 10
C *****
C SET UP INDICES AND COMPILE SUMS
C *****
I=IJS(M)
I=I-1
SMO=SMO+PHIX(I)**2/2.
+PHIX(I)*PHIX(I)
- PHIX(I)**2/2.0-PHIX(I)*PHX(I)
1 SMT=SMT-PHIT(I)**2/2.
2 -PHT(I)*PHIT(I)
+PHIT(I)**2/2.0+PHIT(I)*PHT(I)
1 SMR=(SMR+PHIX(I)*PHIT(I))
2 +PHX(I)*PHIT(I)+PHIX(I)*PHT(I)
+PHX(I)*PHIT(I)+PHX(I)*PHT(I)
2 SMF=SMF-PHI(I)**2/2.
+PHI(I)**2/2.0
GO TO 10
C *****
C THIS SECTION HANDLES CASES WHERE MODE 0 IS INCLUDED
C *****
20 DO 21 L=1,MNMAXO
SMO=SMO+PHIX(L)**2
+2.0*PHX(L)*PHIX(L)
1 SMT=SMT+PHIT(L)**2
+2.0*PHT(L)*PHIT(L)
21 SMF=SMF+PHIX(M)**2
+2.0*PHX(M)*PHIX(M)
1 SMF=SMF+PHI(L)**2
SMT=SMT+PHIT(L)*(PHIT(L)+2.0*PHT(L))
SMR=SMR+(PHIX(L)*(PHIT(L)+PHX(L))+PHX(L)*PHIT(L)+2.0
DO 122 L=2,MNMAXO,2
SMR=SMR+PHIX(L)*(PHIT(L+1)+PHT(L+1)+PHX(L)*PHIT(L+1)
DO 121 L=3,MNMAXO,2
121 SMR=SMR+PHIX(L)*(PHIT(L-1)+PHT(L-1)+PHX(L)*PHIT(L-1)
C *****
C COMPILE BETA TERMS FOR THIS SPECIAL CASE
C *****
BX3(M)=SMO*.5
BT3(M)=SMT*.5
BE3(M)=SMF*.5
BXT3(1)=SMR*.5
GO TO 9
C *****
C THIS SECTION HANDLES ASYMMETRIC MODES
C *****

```



```

C ** 101 MP=M+1 ** 0002480
C ** MAXL=MAXS(MP) ** 0002490
C ** ** 0002500
C ** TEST FOR PRESENCE OF SUMMATION COMBINATIONS ** 0002510
C ** ** 0002520
C ** IF (MAXL.EQ.0) GO TO 102 ** 0002530
C ** ** 0002540
C ** SET UP INDICES FOR SUMMATION COMBINATIONS ** 0002550
C ** ** 0002560
C ** DO 103 L=1,MAXL ** 0002570
C ** I=IS(L,MP) ** 0002580
C ** J=JS(L,MP) ** 0002590
C ** II=I-1 ** 0002600
C ** JJ=J-1 ** 0002610
C ** ** 0002620
C ** TEST FOR MODE 1 AND COMPILE SUMS ** 0002630
C ** ** 0002640
C ** IF (I.EQ.1) GO TO 103 ** 0002650
C ** SMO=SMO+PHX(I)*(PHX(JJ)+PHX(JJ))*(PHX(II)+ ** 0002660
C ** PHX(II))+PHX(II)*PHX(JJ)*PHX(I) ** 0002670
C ** 1 SMT=SMT+PHIT(I)*(PHIT(JJ)+PHIT(JJ))*(PHIT(II)+ ** 0002680
C ** PHIT(II))+PHIT(II)*PHIT(JJ)*PHIT(I) ** 0002690
C ** 1 SMR=SMR+PHX(I)*(PHIT(JJ)+PHIT(JJ))+PHX(JJ)*(PHIT(II)+ ** 0002700
C ** PHIT(II))-PHX(II)*(PHIT(JJ)+PHIT(JJ))-PHX(JJ)* ** 0002710
C ** PHIT(II)+PHIT(I)*PHIT(JJ)+PHX(JJ)*PHIT(II)- ** 0002720
C ** PHX(II)*PHIT(I)-PHX(JJ)*PHIT(I) ** 0002730
C ** SMF=SMF+PHI(I)*PHI(JJ)+PHI(JJ)*PHI(II) ** 0002740
C ** 103 CONTINUE ** 0002750
C ** ** 0002760
C ** TEST FOR PRESENCE OF DIFFERENCE COMBINATIONS ** 0002770
C ** ** 0002780
C ** 102 MAXL=MAXD(MP) ** 0002790
C ** IF (MAXL.EQ.0) GO TO 104 ** 0002800
C ** ** 0002810
C ** SET UP INDICES FOR DIFFERENCE COMBINATIONS ** 0002820
C ** ** 0002830
C ** DO 105 L=1,MAXL ** 0002840
C ** I=ID(L,MP) ** 0002850
C ** J=JD(L,MP) ** 0002860
C ** II=I-1 ** 0002870
C ** JJ=J-1 ** 0002880
C ** ** 0002890
C ** TEST FOR MODE 1 AND COMPILE SUMS ** 0002900
C ** ** 0002910
C ** IF (J.EQ.1) GO TO 123 ** 0002920
C ** SMO=SMO-PHX(I)*(PHX(JJ)+PHX(JJ))+PHX(II)*(PHX(II)+ ** 0002930
C ** PHX(II))+PHX(II)*PHX(JJ)-PHX(JJ)*PHX(II) ** 0002940
C ** 1 ** 0002950

```



```

SMT=SMT+PHIT(I)*(PHIT(JJ)+PHT(JJ))-PHIT(J)*(PHIT(II)+
1 PHT(II))-PHIT(II)*(PHIT(JJ)+PHT(JJ))*PHT(I)
1 SMR=SMR+PHIX(I)*(PHIT(JJ)+PHT(JJ))+PHIX(J)*(PHIT(II)+
1 PHT(II))+PHIX(I)*(PHIT(JJ)+PHT(JJ))+PHIX(J)*(PHIT(II)+
2 (PHIT(II)+PHT(I))+PHIX(I))*PHIT(JJ)+PHIX(J)*PHIT(II)+
3 PHX(II))*PHIT(JJ)+PHIX(JJ)*PHIT(I)
SMF=SMF+PHI(I)*PHI(JJ)-PHI(J)*PHI(II)
GO TO 105
C**
C**
C**
EXECUTE BELOW IF J=1 IN DIFFERENCE COMBINATIONS
C**
C**
123 SMO=SMT+(PHIX(I)*(PHIX(II)+PHX(II))+PHIX(II)*PHX(I))*2.0
SMT=SMT+(PHIT(I)*(PHIT(II)+PHT(I))+PHIT(I)*PHT(II))*2.0
SMR=SMR+(PHIX(I)*(PHIT(II)+PHT(II))+PHIX(II)*PHIT(I)+
1 PHT(I))+PHX(I)*PHIT(II)+PHX(II)*PHIT(I))*2.0
1 SMF=SMF+(PHI(I)*PHI(II))*2.0
105 CONTINUE
C**
C**
C**
TEST FOR PRESENCE OF SAME-INDEX COMBINATION
C**
C**
104 IF (MAXSY(MP).EQ.0) GO TO 10
C**
C**
C**
SET UP INDICES AND COMPILE SUMS
C**
C**
C**
I=IJS(MP)
II=I-1
SMO=SMT+PHIX(I)*(PHIX(II)+PHX(II))+PHIX(II)*PHX(I)
SMT=SMT+PHIT(I)*(PHIT(II)+PHT(II))+PHIT(II)*PHT(I)
SMR=SMR+PHIX(I)*(PHIT(II)+PHT(II))-PHIX(II)*(PHIT(I)+
1 PHT(I))+PHX(I)*PHIT(II)-PHX(II)*PHIT(I)
1 SMF=SMF+PHI(I)*PHI(II)
C**
C**
C**
COMPILE BETA TERMS
C**
C**
C**
10 BX3(M)=SMO
BT3(M)=SMT
BXT3(M)=SMR*.5
BE3(M)=SMF
9 CONTINUE
2222 RETURN
END
00002960
00002970
00002980
00002990
00003000
00003010
00003020
00003030
00003040
00003050
00003060
00003070
00003080
00003090
00003100
00003110
00003120
00003130
00003140
00003150
00003160
00003170
00003180
00003190
00003200
00003210
00003220
00003230
00003240
00003250
00003260
00003270
00003280
00003290
00003300
00003310
00003320
00003330
00003340
00003350
00003360
00003370

```



```

SUBROUTINE MODES
C*****
C IN THIS SUBROUTINE, ARRAYS THAT DEFINE THOSE SETS OF INDICES
C THAT COMBINE TO EQUAL EACH VALUE OF N IN THE PROBLEM ARE DETER-
C MINED. EACH FOURIER INDEX IS CALLED A SPECIFIED INDEX. THE FIRST ITERATION
C REACHED. EACH FOURIER INDEX IS CALLED A SPECIFIED INDEX. THE FIRST ITERATION
C ALL OTHER INDICES TO SEE IF THE NEW VALUE EXISTS IN THE PROGRAM.
C IF IT DOES, THE LOCATION OF THE NEW VALUE IS COMPARED WITH ALL
C BINARIES. THE LOCATION OF THE NEW VALUE IS COMPARED WITH ALL
C JD. ONE ARGUMENT OF THE SPECIAL 2-DIMENSIONAL ARRAYS, ID AND
C ALSO GIVE THE VALUES OF THE NEW COORDINATES. IF THERE IS NO INDEX IN THE
C PROGRAM THAT MATCHES THE NEW ONE, THEN A NEW FOURIER TERM HAS
C BEEN GENERATED AND WILL BE STORED IN THE NEXT ITERATION FOR
C SOLUTION. THE VARIABLE MAXD CONSIDERS THE TOTAL NUMBER OF SUCH
C COMBINATIONS. THE VARIABLE MAXD CONSIDERS THE TOTAL NUMBER OF SUCH
C IN A SIMILAR MANNER, EACH INDEX IS ADDED TO EVERY OTHER
C INDEX AND THE SUM COMPARED WITH ALL INDICES. 'JS', IN THE SAME
C MANNER AS WAS DONE FOR THE SUBTRACTION CASE. COMBINATIONS FOR
C *MAXS, STORED IN THE TOTAL NUMBER OF SUMMATION
C EACH VALUE OF THE FOURIER INDEX. THE CASE WHERE THE INDEX IS ADD-
C ED TO ITSELF, THE SUM COMPARED WITH ALL OTHER INDICES. THE 1-
C DIMENSIONAL ARRAY 'IJS' STORES THE LOCATION OF THE INDEX AND
C THE VARIABLE 'MAXSY' INDICATES IF THIS COMBINATION EXISTS.
C SINCE THE INDEX STATEMENTS ARE REQUIRED IN THAT CASE, ALWAYS GIVES MODE '0',
C NO SPECIFIC INDEX STATEMENTS ARE REQUIRED IN THAT CASE. ALWAYS GIVES MODE '0',
C THE BETA'S AND ALPHA'S CONTAIN NO ZERO TERM, AND THE SUMMATION
C IS CARRIED OUT IN PHIBET(K) AND TEAETA(K) OVER SPECIFICALLY
C DEFINED LIMITS.
C CN INPUT CARDS, INDICATING PROCESSING OF GENERAL ASYMMETRIC
C LOADS AND IMPERFECTIONS. IN SATANS-I, THE NUMBER OF MODES IS
C LIMITED TO 25.
C*****
C COMMON /IBL1/ MNMAX
C /IBL2/ N(25), MNINIT
C*****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410

```



```

C ***** DO WE WANT ANY MORE NEW MODES ***** 00000900
C ***** IF(ICORFL.EQ.1) GO TO 1 ***** 00000910
C ***** FROM HERE TO 1, WE INCREMENT COUNTERS AND MAKE APPROPRIATE ***** 00000920
C ***** ENTRIES IN 'MAXD', 'ID', & 'JD'. TO SIMPLIFY USAGE OF 'ID' ***** 00000930
C ***** AND 'JD' IN TEAETA AND PHIBET. MODES ARE PLACED IN THESE ***** 00000940
C ***** ARRAYS SUCH THAT THE HIGHER MODE NUMBER IS ALWAYS FIRST ***** 00000950
C ***** MNMAX=MNMAX+JUMP ***** 00000960
C ***** N(MNMAX)=NTEST ***** 00000970
C ***** IF (JUMP.GT.1) N(MNMAX-1)=-NTEST ***** 00000980
C ***** MMFT=MNMAX ***** 00000990
C ***** IF(MNMAX.EQ.MAXM) ICORFL=1 ***** 0001000
C ***** IF(NMN-MNM) 11,1,12 ***** 0001010
C ***** 10 LOC D=MAXD(MMFT)+1 ***** 0001020
C ***** 11 MAXD(MMFT)=LOC D ***** 0001030
C ***** ID(LOC D,MMFT)=MM ***** 0001040
C ***** JD(LOC D,MMFT)=MN ***** 0001050
C ***** GO TO 1 ***** 0001060
C ***** 12 LOC D=MAXD(MMFT)+1 ***** 0001070
C ***** MAXD(MMFT)=LOC D ***** 0001080
C ***** ID(LOC D,MMFT)=MN ***** 0001090
C ***** JD(LOC D,MMFT)=MM ***** 0001100
C ***** 1 CONTINUE ***** 0001110
C ***** ***** 0001120
C ***** ***** 0001130
C ***** ***** 0001140
C ***** ***** 0001150
C ***** ***** 0001160
C ***** ***** 0001170
C ***** ***** 0001180
C ***** ***** 0001190
C ***** ***** 0001200
C ***** ***** 0001210
C ***** ***** 0001220
C ***** ***** 0001230
C ***** ***** 0001240
C ***** ***** 0001250
C ***** ***** 0001260
C ***** ***** 0001270
C ***** ***** 0001280
C ***** ***** 0001290
C ***** ***** 0001300
C ***** ***** 0001310
C ***** ***** 0001320
C ***** ***** 0001330
C ***** ***** 0001340
C ***** ***** 0001350
C ***** ***** 0001360
C ***** ***** 0001370

      THIS SECTION HANDLES MODES GENERATED FROM THE SUMMATION
      COMBINATIONS OF EXISTING MODES

      NOW SET UP INITIAL LOOP FOR SUM COMBINATIONS
      DO 301 MN=1,MNMAXO,JUMP
      NMN=N(MN)
      NNS=MN
      IF(MNINIT.GT.MN) NNS=MNINIT
      SET UP SECOND LOOP - TO ADD PRESENT MODE TO ALL OTHER MODES
      DO 301 MM=NNS,MNMAXO,JUMP
      NMM=N(MM)
      CREATE SUM
      NTEST=NMN+NMM

```



```

C ** ** ** ** ** ** ** ** ** ** ** ** ** ** **  00001380
C ** ** **  TO COMPARE SUM WITH ALL OTHER MODES  ** ** **  00001390
C ** ** **  DO 302 MMFT=1,MNMAX,JUMP  ** ** **  00001400
C ** ** **  IF WE SATISFY HERE, MODE EXISTS, GO TO INCREMENT ,MAXS, (OR  ** ** **  00001420
C ** ** **  ,MAXSY,.)  ** ** **  00001430
C ** ** **  IF(NTEST.EQ.N(MMFT)) GO TO 310  ** ** **  00001440
C ** ** **  302 CONTINUE  ** ** **  00001450
C ** ** **  IF WE MAKE IT TO HERE, WE HAVE GENERATED A NEW MODE  ** ** **  00001460
C ** ** **  DO WE WANT ANY MORE NEW MODES  ** ** **  00001470
C ** ** **  IF(ICORFL.EQ.1) GO TO 301  ** ** **  00001480
C ** ** **  IF(MNMAX.GE.MAXM) GO TO 301  ** ** **  00001490
C ** ** **  INCREMENT ,MNMAX, AND ESTABLISH NEW MODE NUMBER  ** ** **  00001500
C ** ** **  MNMAX=MNMAX+JUMP  ** ** **  00001510
C ** ** **  N(MNMAX)=NTEST  ** ** **  00001520
C ** ** **  IF (JUMP.GT.1) N(MNMAX-1)=-NTEST  ** ** **  00001530
C ** ** **  MMFT=MNMAX  ** ** **  00001540
C ** ** **  IF(MNMAX.GE.MAXM) ICORFL=1  ** ** **  00001550
C ** ** **  IF MODE WAS ADDED TO ITSELF, GO TO ,MAXSY & IJS, SECTION  ** ** **  00001560
C ** ** **  310 IF(MN.EQ.MM) GO TO 360  ** ** **  00001570
C ** ** **  MAKE ENTRIES IN ,LOCS, ,IS, & ,JS,  ** ** **  00001580
C ** ** **  LOCS=MAXS(MMFT)+1  ** ** **  00001590
C ** ** **  MAXS(MMFT)=LOCS  ** ** **  00001600
C ** ** **  IS(LOCS,MMFT)=MN  ** ** **  00001610
C ** ** **  JS(LOCS,MMFT)=MM  ** ** **  00001620
C ** ** **  GO TO 301  ** ** **  00001630
C ** ** **  SEE IF THE SUM OF THE MODE WITH ITSELF WAS THE 0-TH MODE  ** ** **  00001640
C ** ** **  360 IF(MN.EQ.0) GO TO 301  ** ** **  00001650
C ** ** **  IF HERE, IT WASN'T, MAKE ENTRIES IN ,MAXSY, AND ,IJS,  ** ** **  00001660
C ** ** **  MAXSY(MMFT)=1  ** ** **  00001670
C ** ** **  IJS(MMFT)=MN  ** ** **  00001680
C ** ** **  MNINIT=MNMAXO+JUMP  ** ** **  00001690
C ** ** **  301 CONTINUE  ** ** **  00001700
C ** ** **  ** ** **  00001710
C ** ** **  ** ** **  00001720
C ** ** **  ** ** **  00001730
C ** ** **  ** ** **  00001740
C ** ** **  ** ** **  00001750
C ** ** **  ** ** **  00001760
C ** ** **  ** ** **  00001770
C ** ** **  ** ** **  00001780
C ** ** **  ** ** **  00001790
C ** ** **  ** ** **  00001800
C ** ** **  ** ** **  00001810
C ** ** **  ** ** **  00001820
C ** ** **  ** ** **  00001830
C ** ** **  ** ** **  00001840
C ** ** **  ** ** **  00001850

```


00001860
00001870
00001880
00001890

```
IF(ICORFL.GT.0) IPASS=IPASS+1
IF(IPASS.LT.2.AND.MNINIT.LE.MNMAX) CALL PMATRIX
RETURN
END
```

```
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320

SUBROUTINE INLPOL
  THIS SUBROUTINE COMPUTES THE NON-LINEAR TERMS BETA-SUB S,
  -SUB THETA, -SUB S-THETA, AETA-SUB S-S AND -SUB THETA-S AT AN
  INITIAL POLE.
  COMMON /IBL1/ MNMAX
  COMMON /IBL3/ M0,M1,M2,M3
  COMMON /IBL11/ ICORFL,IPASS,JUMP,MPERFS
  COMMON /IBL12/ KMAX1,KMAX2,NCONV
  COMMON /IBL13/ ITRMAX,LSMAX
  COMMON /BL5/ TT(25),EMT(25),DT(25),DMT(25)
  COMMON /BL6/ Z(4,1001),SOE,OSE,ALOAD
  COMMON /BL7/ DL,S1
  COMMON /BL11/ OMXI(100),PHEE,T0,T2
  COMMON /BL11A/ PHEN,T2N
  COMMON /BL17/ DEL
  COMMON /BL29/ BX1(25),BT1(25),BX2(25),BT2(25),
  1 BX2(25),BE2(25)
  1 EXX1(25),ET1(25),EX1(25),ET1(25),EXX2(25),
  1 ET2(25),EXT2(25),EX2(25),ET2(25)
  1 DELSQ,EXT1(25)
  IF (JUMP.EQ.2) GO TO 1000
  DO 1 MN=1,MNMAX
    BX1(MN)=0.
    BT1(MN)=0.
    BX11(MN)=0.
    BE1(MN)=0.
    EX1(MN)=0.
    ET1(MN)=0.
    ETX1(MN)=0.
  
```



```

1 EXX1(MN)=0.
  IF(M1.EQ.0) RETURN
  I2=2+(M1-1)*KMAX2
  I3=I2+1
  I4=I3+1
  PHEE=(1.5*Z(3,I2)-2.*Z(3,I3)+.5*Z(3,I4))/DEL+OMXI(1)*Z(1,I2)
  BET=.5*PHEE**2
  IF(IIRMAX.EQ.1) BET=0.
  T2=0.
  IF(M2.EQ.0) GO TO 2
  CALL BDB(1,B,DB,D,DD)
  I2=2+(M2-1)*KMAX2
  I3=I2+1
  I4=I3+1
  T2=B*DI*((-1.5*Z(1,I2)+2.*Z(1,I3)-.5*Z(1,I4))/DEL+.5*SOE*BET)
  Q1=.5*PHEE*T2
  BX1(M2)=BET
  BT1(M2)=-BET
  BXT1(M2)=-BET
  ETX1(M1)=Q1 GO TO 2
  IF(M3.EQ.0) GO TO 2
  EXX1(M3)=Q1
  ETX1(M3)=-Q1
  T2=0.
  IF(M0.EQ.0) GO TO 3
  BX1(M0)=BET
  BT1(M0)=BET
  CALL BDB(1,B,DB,D,DD)
  CALL TLOAD(1)
  I2=2+(M0-1)*KMAX2
  I3=I2+1
  I4=I3+1
  T2=B*SI*((-1.5*Z(1,I2)+2.*Z(1,I3)-.5*Z(1,I4))/DEL+OMXI(1)*Z(3,I2)
  1+.5*SOE*BET)-TT(M0)*ALOAD
3 EXX1(M1)=PHEE*(T2+.5*T2)
  RETURN
1000 DO 1001 MN=1,MNMAX
      BX1(MN)=0.
      BT1(MN)=0.
      BXT1(MN)=0.
      BET(MN)=0.
      EX1(MN)=0.
      ET1(MN)=0.
      ETX1(MN)=0.
      EXX1(MN)=0.
      IF(M1.EQ.0) RETURN
      I2=2+(M1-1)*KMAX2
      I3=I2+1

```



```

I4=I3+1
PHEE=(1.5*Z(3,I2)-2.*Z(3,I3)+.5*Z(3,I4))/DEL+OMXI(1)*Z(1,I2)
T2=0.
IF(M2.EQ.0) GO TO 1002
CALL BDB(1,B,DB,D,DD)
I2=2+(M2-1)*KMAX2
I3=I2+1
I4=I3+1
PHX1=PHIXB(KMAX+1)
PHX2=PHIXB(2*KMAX+1)
PHEN=(1.5*Z(3,I2-KMAX2)-2.*Z(3,I3-KMAX2)+.5*Z(3,I4-KMAX2))/DEL+
1 OMXI(1)*Z(1,I2-KMAX2)
BX1(M2)=.5*(PHEE*(PHEN+2.*PHX1)-PHEN*(PHEN+2.*PHX2))
IF(ITRMAX.EQ.1) BX1(M2)=0.
BT1(M2)=-BX1(M2)
BX1(M2)=-BX1(M2)
T2=B*D1*((-1.5*Z(1,I2)+2.*Z(1,I3)-.5*Z(1,I4))/DEL+.5*SOE*BX1(M2))
M2L=M2-1
BX1(M2L)=PHEE*(PHEN+PHX2)+PHX1*PHEN
IF(ITRMAX.EQ.1) BX1(M2L)=0.
BT1(M2L)=-BX1(M2L)
BX1(M2L)=BX1(M2L)
T2N=B*D1*((-1.5*Z(1,I2-KMAX2)+2.*Z(1,I3-KMAX2)-.5*Z(1,I4-KMAX2))
1 /DEL+.5*SOE*BX1(M2L))
1002 T2=0.
IF(M2.EQ.0) GO TO 1003
BX1(M2)=.5*(PHEE*(PHEN+2.*PHX1)+PHEN*(PHEN+2.*PHX2))
IF(ITRMAX.EQ.1) BX1(M2)=0.
BT1(M2)=BX1(M2)
CALL BDB(1,B,DB,D,DD)
CALL TLOAD(1)
I2=2+(M2-1)*KMAX2
I3=I2+1
I4=I3+1
T2=B*D1*((-1.5*Z(1,I2)+2.*Z(1,I3)-.5*Z(1,I4))/DEL+OMXI(1)*Z(3,I2)
1 +.5*SOE*BX1(M2))-T1(M2)*ALOAD
1003 IF(ITRMAX.EQ.1) RETURN
PHSS=PHEN+PHX1
PHSP=PHEN+PHX2
M1=M1-1
EXX1(M1)=PHSS*T2+.5*(PHSS*T2+PHSP*T2N)
EXX1(M1)=PHSP*T2-.5*(PHSP*T2-PHSS*T2N)
ETX1(M1)=.5*(PHSS*T2+PHSP*T2N)
ETX1(M1)=.5*(-PHSP*T2+PHSS*T2N)
IF(M3.EQ.0) RETURN
M3L=M3-1
EXX1(M3)=.5*(PHSS*T2-PHSP*T2N)
EXX1(M3L)=.5*(PHSS*T2N+PHSP*T2)

```


00001290
00001300
00001310
00001320

```
ETX1(M3)=.5*(-PHSS*T2+PHSP*T2)
ETX1(M3L)=.5*(-PHSP*T2-PHSS*T2N)
RETURN
END
```

```

SUBROUTINE FNLPOL
C*****
C THIS SUBROUTINE COMPUTES THE NON-LINEAR TERMS BETA-SUB S,
C -SUB THETA, -SUB S-THETA, AETA-SUB S-S, AND -SUB THETA-S AT A
C FINAL POLE.
C*****
COMMON /IBL1/ MNMAX
COMMON /IBL3/ M3,M1,M2,M3
COMMON /IBL4/ KMAX,KL
COMMON /IBL11/ ICORFL,IPASS,JUMP,MPERFS
COMMON /IBL12/ KMAX1,KMAX2,NCONV
COMMON /IBL13/ ITRMAX,LSMAX
COMMON /BL5/ TT(25),EMT(25),DT(25),DMT(25)
COMMON /BL6/ Z(4,1001),SOE,OSE,ALOAD
COMMON /BL7/ D1,S1
COMMON /BL11A/ PHEN,T2N
COMMON /BL11/ OMXI(100),PHEE,TO,T2
COMMON /BL17/ DEL
COMMON /BL28/ BX3(25),BT3(25),EXT3(25),ETX3(25),EX3(25),ET3(25)
C*****
IF (JUMP.EQ.2) GO TO 1000
DO 1 MN=1,MNMAX
BX3 (MN)=0.
BT3 (MN)=0.
EXT3 (MN)=0.
BE3 (MN)=0.
EX3 (MN)=0.
ET3 (MN)=0.
ETX3 (MN)=0.
EXX3 (MN)=0.
1 CALL BDB(KMAX,B,DB,D,DB)
C*****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320

```



```

IF(M1.EQ.0) RETURN
KM=KMAX1+(M1-1)*KMAX2
KM1=KM-1
KM2=KM-2
PHEE=-(1.5*Z(3,KM)-2.*Z(3,KM1)+.5*Z(3,KM2))/DEL+OMXI(KMAX)*Z(1,KM)
BET=.5*PHEE#2
IF(ITRMAX.EQ.1) BET=0.
T2=0.
IF(M2.EQ.0) GO TO 2
KM=KMAX1+(M2-1)*KMAX2
KM1=KM-1
KM2=KM-2
T2=B#D1*((1.5*Z(1,KM)-2.*Z(1,KM1)+.5*Z(1,KM2))/DEL+.5*SOE*BET)
Q1=.5*PHEE#T2
BX3(M2)=-BET
BXT3(M2)=BET
ETX3(M1)=Q1
IF(M3.EQ.0) GO TO 2
EXX3(M3)=Q1
ETX3(M3)=-Q1
TJ=J.
2 IF(M0.EQ.0) GO TO 3
CALL TLOAD(KMAX)
KM=KMAX1+(M0-1)*KMAX2
KM1=KM-1
KM2=KM-2
BX3(M0)=BET
BT3(M0)=BET
TO=B#S1*((1.5*Z(1,KM)-2.*Z(1,KM1)+.5*Z(1,KM2))/DEL+OMXI(KMAX)*
1 Z(3,KM)+.5*SOE*BET)-TT(M0)*ALOAD
3 EXX3(M1)=PHEE*(TO+.5*T2)
RETURN
1000 DO 1001 MN=1,MNMAX
BX3(MN)=0.
BT3(MN)=0.
BXT3(MN)=0.
BE3(MN)=0.
EX3(MN)=0.
ET3(MN)=0.
ETX3(MN)=0.
EXX3(MN)=0.
CALL BDB(KMAX,B,DB,D,DB)
IF(M1.EQ.0) RETURN
KM=KMAX1+(M1-1)*KMAX2
KM1=KM-1
KM2=KM-2
PHEE=-(1.5*Z(3,KM)-2.*Z(3,KM1)+.5*Z(3,KM2))/DEL+OMXI(KMAX)*Z(1,KM)
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800

```



```

T2=0.
IF(M2.EQ.0) GO TO 1002
KM=KMAX1+(M2-1)*KMAX2
KM1=KM-1
KM2=KM-2
J=KMAX#2
I=J+KMAX
PHX1=PHI XB(J)
PHX2=PHI XB(I)
PHEN=- (1.5#Z(3,KM-KMAX2)-2.*Z(3,KM1-KMAX2)+.5*Z(3,KM2-KMAX2))/DEL
1+OMXI(KMAX)*Z(1,KM-KMAX2)
BX3(M2)=.5*(PHEE*(PHEN+2.*PHX1)-PHEN*(PHEN+2.*PHX2))
IF(ITRMAX.EQ.1) BX3(M2)=0.
BT3(M2)=-BX3(M2)
BX3(M2)=BX3(M2)
M2L=M2-1
BX3(M2L)=PHEE*(PHEN+PHX2)+PHX1*PHEN
IF(ITRMAX.EQ.1) BX3(M2L)=0.
BT3(M2L)=-BX3(M2L)
BX3(M2L)=-BX3(M2L)
T2=B#D1*((1.5#Z(1,KM)-2.*Z(1,KM1)+.5*Z(1,KM2))/DEL+.5*SOE*B#X3(M2))
T2N=B#D1*((1.5#Z(1,KM-KMAX2)-2.*Z(1,KM1-KMAX2)+.5*Z(1,KM2-KMAX2))/
1 /DEL+.5*SOE*B#X3(M2L))
1002 TO=0.
IF(M3.EQ.0) GO TO 1003
CALL TLOAD(KMAX)
KM=KMAX1+(M0-1)*KMAX2
KM1=KM-1
KM2=KM-2
BX3(M0)=.5*(PHEE*(PHEE+2.*PHX1)+PHEN*(PHEN+2.*PHX2))
IF(ITRMAX.EQ.1) BX3(M0)=0.
BT3(M0)=BX3(M0)
T3=B#S1*((1.5#Z(1,KM)-2.*Z(1,KM1)+.5*Z(1,KM2))/DEL+OMXI(KMAX)*
1 Z(3,KM)+.5*SOE*B#X3(M0))-TT(M0)*ALOAD
1003 IF(ITRMAX.EQ.1) RETURN
PHSS=PHEE+PHX1
PHSP=PHEN+PHX2
M1L=M1-1
EXX3(M1)=PHSS*T0+.5*(PHSS*T2+PHSP*T2N)
EXX3(M1L)=PHSP*T0-.5*(PHSP*T2-PHSS*T2N)
ETX3(M1)=.5*(PHSS*T2+PHSP*T2N)
ETX3(M1L)=.5*(-PHSP*T2+PHSS*T2N)
IF(M3.EQ.0) RETURN
M3L=M3-1
EXX3(M3)=.5*(PHSS*T2-PHSP*T2N)
EXX3(M3L)=.5*(PHSS*T2N+PHSP*T2)
ETX3(M3)=.5*(-PHSP*T2+PHSP*T2N)
ETX3(M3L)=.5*(-PHSP*T2-PHSS*T2N)
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900
00000910
00000920
00000930
00000940
00000950
00000960
00000970
00000980
00000990
00001000
00001010
00001020
00001030
00001040
00001050
00001060
00001070
00001080
00001090
00001100
00001110
00001120
00001130
00001140
00001150
00001160
00001170
00001180
00001190
00001200
00001210
00001220
00001230
00001240
00001250
00001260
00001270
00001280

```


RETURN
END

00001290
00001300

```

C***** SUBROUTINE POLE(K) *****
C***** THIS SUBROUTINE PRINTS THE SOLUTION AT AN INITIAL AND A FINAL *****
C***** POLE. *****
C***** IMPLICIT LOGICAL*1 ($) *****
REAL NU, MT, MX, MTH, MXT, MTS, KX, KT, KXT, LAM, LAM2, MASS
COMMON /IBL2/ N(25), MNINIT
COMMON /IBL3/ MO, M1, M2, M3
COMMON /IBL4/ KMAX, KL, IBCFNL
COMMON /IBL5/ MNMAXO, MAXD(25), MAXS(25), MAXSY(25), IS(25, 25),
COMMON /IBL7/ JS(25, 25), ID(25, 25), JD(25, 25), IJS(25)
1 COMMON /IBL8/ LSTEP, ITR, HMAX
COMMON /IBL10/ IFREQ, NTHMAX
COMMON /IBL11/ ICORFL, IPASS, JUMP, MPERFS
COMMON /IBL12/ KMAX1, KMAX2, NCONV
COMMON /BL4/ P(4, 4, 951), X(4, 951), ZF1M(4, 4, 25), ZF2M(4, 4, 25),
COMMON /BL5/ ZF3M(4, 4, 25), ZF4M(4, 4, 25)
COMMON /BL6/ TT(25), MT(25), DT(25), DMT(25)
COMMON /BL7/ Z(4, 100), SOE, OSE, ALOAD
COMMON /BL8/ DI, SI
COMMON /BL10/ R(100), GAM(100), OMT(100)
COMMON /BL11/ PHIX(25), PHIT(25), PHI(25)
COMMON /BL11/ OMXI(100), PHEE, T0, T2
COMMON /BL11A/ PHEN, T2N
COMMON /BL12/ TDLI, TDEL
COMMON /BL14/ LAM2, LSD18, LSD1N
COMMON /BL15/ NU, UI(25), VI(25), W1(25), V2(25), U2(25), W2(25), U3(25),
COMMON /BL15/ V3(25), W3(25)
1 COMMON /BL17/ DEL
COMMON /BL19/ TH(36)
COMMON /BL20/ DEOMX(100)
COMMON /BL27/ BX3(25), BT3(25), BXT3(25), BE3(25)

```



```

COMMON /BL31/ DELSQ,EXT1(25)
COMMON /BL32/ TKN,ELAST,CHAR,SIGO
COMMON /BL100/ TEEO,$DYNMC
COMMON /BL101/ ZO(4,1001),Z2(4,1001),Z3(4,1001),DELSO
COMMON /BL102/ DELOAD
COMMON /BL103/ MASS(100)
COMMON /BL110/ TX(25),TTH(25),MX(25),MTH(25),MXT(25),
1
COMMON /BL111/ ABZ,ABZO,ABZN,ABZ3,DD2
C*****
IF (JUMP.EQ.2) GO TO 1000
CALL BDB(K,BS,DB,DS,DD)
IF(K.EQ.KMAX) GO TO 301
DO 202 MN=1,MNMAXO
U1(MN)=U2(MN)
V1(MN)=V2(MN)
W1(MN)=W2(MN)
I3=3+(MN-1)*KMAX2
I2=I3-1
U2(MN)=Z(1,I3)
V2(MN)=Z(2,I3)
W2(MN)=Z(3,I3)
PHIX(MN)=0.0
PHIT(MN)=0.0
PHI(MN)=0.0
MX(MN)=Z(4,I2)*ABZ3
MXT(MN)=0.0
QS(MN)=0.0
TX(MN)=0.0
TTH(MN)=0.0
TXT(MN)=0.0
IF(M1.EQ.0) GO TO 203
CALL INLPOL
PHIX(M1)=PHEE*ABZO
PHIT(M1)=-PHEE*ABZO
QS(M1)=0.0
IF(MJ.EQ.0) GO TO 204
TX(MO)=TO*ABZ
TTH(MO)=TO*ABZ
MTH(MO)=MX(MO)
IF(M2.EQ.0) GO TO 205
TX(M2)=T2*ABZ
TTH(M2)=-T2*ABZ
TXT(M2)=-T2*ABZ
MTH(M2)=-MX(M2)
MXT(M2)=MTH(M2)
GO TO 205
202
IF(M1.EQ.0) GO TO 203
CALL INLPOL
PHIX(M1)=PHEE*ABZO
PHIT(M1)=-PHEE*ABZO
QS(M1)=0.0
IF(MJ.EQ.0) GO TO 204
TX(MO)=TO*ABZ
TTH(MO)=TO*ABZ
MTH(MO)=MX(MO)
IF(M2.EQ.0) GO TO 205
TX(M2)=T2*ABZ
TTH(M2)=-T2*ABZ
TXT(M2)=-T2*ABZ
MTH(M2)=-MX(M2)
MXT(M2)=MTH(M2)
GO TO 205
204
IF(M1.EQ.0) GO TO 203
CALL INLPOL
PHIX(M1)=PHEE*ABZO
PHIT(M1)=-PHEE*ABZO
QS(M1)=0.0
IF(MJ.EQ.0) GO TO 204
TX(MO)=TO*ABZ
TTH(MO)=TO*ABZ
MTH(MO)=MX(MO)
IF(M2.EQ.0) GO TO 205
TX(M2)=T2*ABZ
TTH(M2)=-T2*ABZ
TXT(M2)=-T2*ABZ
MTH(M2)=-MX(M2)
MXT(M2)=MTH(M2)
GO TO 205

```



```

203 IF(M0.EQ.0) GO TO 206
I3=3+(M0-1)*KMAX2
I4=I3+1
CALL TLOAD(1)
TX(M0)=BS*SI*((2.*Z(1,I3)-.5*Z(1,I4))/DEL+OMXI(1)*Z(3,I3-1))*ABZ
1 TT(M0)=TX(M0)
MTH(M0)=MX(M0)
206 IF(M2.EQ.0) GO TO 205
I3=3+(M2-1)*KMAX2
I4=I3+1
TX(M2)=BS*D1*((2.*Z(1,I3)-.5*Z(1,I4))/DEL
TX(M2) = TX(M2)*ABZ
TTH(M2) = -TX(M2)
TXT(M2) = -TX(M2)
MTH(M2) = -MX(M2)
MXI(M2) = -MX(M2)
RETURN
205 CONTINUE
301 DO 302 MN=1,MNMAX0
U1(MN)=U2(MN)
V1(MN)=V2(MN)
W1(MN)=W2(MN)
PHIX(MN)=0.
PHIT(MN)=0.
PHI(MN)=0.
IK=KMAX1+(MN-1)*KMAX2
MX(MN)=Z(4,IK)*ABZ3
MTH(MN)=0.
MXI(MN)=0.
QS(MN)=0.
TX(MN)=0.
TTH(MN)=0.
TXT(MN)=0.
302 IF(M1.EQ.0) GO TO 303
CALL FNLPOL
PHIX(M1)=PHEE*ABZ0
PHIT(M1)=PHEE*ABZ0
QS(M1)=0.
IF(M0.EQ.0) GO TO 304
TX(M0)=I0*ABZ
TTH(M0)=I0*ABZ
MTH(M0)=MX(M0)
304 IF(M2.EQ.0) GO TO 305
TX(M2)=I2*ABZ
TTH(M2)=-I2*ABZ
TXT(M2)=I2*ABZ
MTH(M2)=-MX(M2)

```



```

303      MXT(M2) = MX(M2)
      GO TO 305
      IF(MO.EQ.0) GO TO 306
      IKM=KMAX+(MO-1)*KMAX2
      IM1=IKM-1
      CALL TLOAD(KMAX)
      TX(MO)=BS*SI*((-2.*Z(1,IKM)+.5*Z(1,IM1))/DEL+OMXI(KMAX)*Z(3,IKM+1)
1      )*ABZ-TT(MO)*ABZ*ALOAD
      TTH(MO) = TX(MO)
      MTH(MO) = MX(MO)
      IF(M2.EQ.0) GO TO 305
      IKM=KMAX+(M2-1)*KMAX2
      IM1=IKM-1
      TX(M2)=BS*DI*((-2.*Z(1,IKM)+.5*Z(1,IM1))/DEL
      TX(M2)=TX(M2)*ABZ
      TTH(M2) = -TX(M2)
      TXT(M2) = TX(M2)
      MTH(M2) = -MX(M2)
      MXT(M2) = MX(M2)
305      RETURN
1000     CALL BDB(K,BS,DB,DS,DD)
      IF(K.EQ.KMAX) GO TO 1301
      DO 1202 MN=1,MNMAXO
      U1(MN)=U2(MN)
      V1(MN)=V2(MN)
      W1(MN)=W2(MN)
      I3=3+(MN-1)*KMAX2
      I2=I3-1
      U2(MN)=Z(1,I3)
      V2(MN)=Z(2,I3)
      W2(MN)=Z(3,I3)
      PHIX(MN)=0.
      PHIT(MN)=0.
      PHI(MN)=0.
      MX(MN)=Z(4,I2)*ABZ3
      MTH(MN)=0.
      MXT(MN)=0.
      QS(MN)=0.
      TX(MN)=0.
      TTH(MN)=0.
      TXT(MN)=0.
1202     IF(M1.EQ.0) GO TO 1203
      CALL INLPOL
      M1L=M1-1
      M2L=M2-1
      PHIX(M1)=PHEE*ABZO
      PHIX(M1L)=PHEN*ABZO
      PHIT(M1)=--PHEE*ABZO

```



```

PHIT(MIL)=PHIX(MIL)
QS(MI)=0.
QS(MIL)=0.
IF(MO.EQ.0) GO TO 1204
TX(MO)=TJ*ABZ
MTH(MO)=TO*ABZ
MTH(MO)=MX(MO)
1204 IF(M2.EQ.0) GO TO 1205
TX(M2)=T2*ABZ
TTH(M2)=-T2*ABZ
TX(M2)=-T2*ABZ
MTH(M2)=-MX(M2)
MTH(M2)=MTH(M2)
TX(M2L)=T2N*ABZ
TTH(M2L)=-TX(M2L)
TX(M2L)=TX(M2L)
MTH(M2L)=-MX(M2L)
MTH(M2L)=MX(M2L)
GO TO 1205
1203 IF(MO.EQ.0) GO TO 1206
I3=3+(MO-1)*KMAX2
I4=I3+1
CALL TLOAD(1)
TX(MO)=BS*SI*((2.*Z(1,I3)-.5*Z(1,I4))/DEL+OMXI(1)*Z(3,I3-1))*ABZ
1 -IT(MO)*ABZ*ALOAD
TTH(MO)=TX(MO)
MTH(MO)=MX(MO)
1206 IF(M2.EQ.0) GO TO 1205
I3=3+(M2-1)*KMAX2
I4=I3+1
TX(M2)=BS*D1*((2.*Z(1,I3)-.5*Z(1,I4))/DEL
TX(M2)=TX(M2)*ABZ
TTH(M2)=-TX(M2)
TX(M2)=-TX(M2)
MTH(M2)=-MX(M2)
MTH(M2)=-MX(M2)
TX(M2L)=BS*D1*((2.*Z(1,I3-KMAX2)-.5*Z(1,I4-KMAX2))/DEL
TX(M2L)=TX(M2L)*ABZ
TTH(M2L)=-TX(M2L)
TX(M2L)=TX(M2L)
MTH(M2L)=-MX(M2L)
MTH(M2L)=MX(M2L)
1205 RETURN
1301 CONTINUE
DO 1302 MN=1,MNMAXO
U1(MN)=U2(MN)
V1(MN)=V2(MN)
W1(MN)=W2(MN)

```



```

00002750
00002760
00002770
00002780
00002790
00002800
00002810
00002820
00002830
00002840
00002850
00002860

```

```

      ITH(M2) = -TX(M2)
      IXT(M2) = -TX(M2)
      MTH(M2) = -MX(M2)
      MXT(M2) = MX(M2)
      TX(M2L) = BS*DI*(-2.*Z(1,IKM-KMAX2)+.5*Z(1,IM1-KMAX2))/DEL
      TX(M2L) = TX(M2L)*ABZ
      ITH(M2L) = -TX(M2L)
      IXT(M2L) = -TX(M2L)
      MTH(M2L) = -MX(M2L)
      MXT(M2L) = MX(M2L)
      RETURN
      END
1305

```

```

C***** SUBROUTINE OUTPUT(IMODE)
C***** THIS SUBROUTINE PREPARES THE PRINTOUT MATERIAL. EVERY IPRINT *****
C***** CONVERGED SOLUTION IS PRINTED. THE FOURIER COEFFICIENTS OF THE *****
C***** IN PLANE FORCES, MERIDIONAL TRANSVERSE FORCE, CIRCUMFERENTIAL *****
C***** BENDING MOMENT, TWISTING MOMENT AND ROTATIONS CAN BE COMPUTED *****
C***** AND PRINTED WITH THE SOLUTION Z FOR THE FOURIER COEFFICIENTS *****
C***** OF THE THREE DISPLACEMENTS AND MERIDIONAL BENDING MOMENT. THIS *****
C***** OUTPUT MATERIAL IS CONVERTED FROM DIMENSIONLESS FORM TO DIMENSIONS *****
C***** SIONAL FORM HERE. ETC. PUTTING THE TOTAL VALUES OF THE FORCES *****
C***** 1, IFREQ+1, 2IFREQ+1, ETC. PUTTING THE TOTAL VALUES OF THE FORCES *****
C***** SUMMATION OF DISPLACEMENTS AND ROTATIONS AT THE INPUT DATA *****
C***** AROUND THE CIRCUMFERENCE PRESCRIBED IN THE INPUT DATA *****
C***** REAL NJ,MT,MX,MTH,MXT,MTS,KX,KT,KXT,LAM,LAM2,MASS *****
C***** INTEGER N(25),MNINIT *****
C***** COMMON /IBL2/ M0,M1,M2,M3 *****
C***** COMMON /IBL3/ KMAX,KL *****
C***** COMMON /IBL4/ IBCFNL *****
C***** COMMON /IBL5/ MNMAXO,MAXD(25),MAXS(25),MAXSY(25),IS(25,25), *****
C***** COMMON /IBL7/ JS(25,25),ID(25,25),JD(25,25),IJS(25) *****
C*****
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230

```



```

COMMON /IBL8/ LSTEP, ITR
COMMON /IBL10/ IFREQ, NTHMAX
COMMON /IBL11/ ICORFL, IPASS, JUMP, MPERFS
COMMON /IBL12/ KMAX1, KMAX2, NCONV
COMMON /IBL13/ ITRMAX, LSMAX
COMMON /BL4/ P(4,4,951), X(4,951), ZF1M(4,4,25), ZF2M(4,4,25),
1 ZF3M(4,4,25), ZF4M(4,4,25)
COMMON /BL5/ TT(25), MT(25), DT(25), DMT(25)
COMMON /BL6/ Z(4,1001), SOE, OSE, ALOAD
COMMON /BL7/ D1, S1
COMMON /BL8/ R(100), GAM(100), DMT(100)
COMMON /BL10/ PHIX(25), PHIT(25), PHI(25)
COMMON /BL11/ OMXI(100), PHEE, IO, T2
COMMON /BL12/ TDL1, TDEL
COMMON /BL14/ LAM2, LSD18, LSD1N
COMMON /BL15/ NU, U1(25), U2(25), U3(25), V1(25), V2(25), V3(25), W1(25),
1 W2(25), W3(25)
COMMON /BL17/ DEL
COMMON /BL19/ TH(36)
COMMON /BL20/ DEOMX(100)
COMMON /BL27/ BX3(25), BXT3(25), BE3(25)
COMMON /BL31/ DELSQ, EXT1(25)
COMMON /BL32/ TKN, ELAST, CHAR, SIGO
COMMON /BL100/ SORD, TEEO
COMMON /BL101/ ZO(4,1001), Z2(4,1001), Z3(4,1001), DELSD
COMMON /BL102/ DELOAD
COMMON /BL103/ MASS(100)
COMMON /BL110/ TX(25), TTH(25), TXT(25), MX(25), MTH(25), MXT(25),
1 QS(25)
COMMON /BL111/ ABZ, ABZO, ABZN, ABZ3, DD2
DIMENSION PTF(100), PF(100)
C*****
ABZO=SIGO/ELAST
IF(SORD.NE.0) GO TO 181
WRITE(6,101) LSTEP, ALOAD, ITR
GO TO 182
181 TI=LSTEP*DELOAD
TI=TI*TEEO
WRITE(6,151) LSTEP, TI, DTI, ITR
182 LAM=TKN/CHAR
ENL=1.
ABZ=SIGO*TKN
ABZ3=ABZ*TKN*TKN/CHAR
ABZN=CHAR*SIGO/ELAST
IF(IIRMAX.EQ.1) ENL=0.
DD2=1.-NU*2
DDI=1./DD2
DPI=1./S1
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710

```



```

DNI=1./D1/DELSQ
TDLISQI=.5/DELSQ GO TO 991
IF(NTHMAX.EQ.0) GO TO 991
DO 21 NTH=1,NTHMAX
DO 1 MN=1,MNMAXO
I1=1+(MN-1)*KMAX2
I2=I1+1
U1(MN)=Z(1,I1)
U2(MN)=Z(1,I2)
V1(MN)=Z(2,I1)
V2(MN)=Z(2,I2)
W1(MN)=Z(3,I1)
W2(MN)=Z(3,I2)
TTH=TH(NTH)
WRITE(6,I16) TTH
DO 121 K=1,KMAX
K1=K+1
CALL BDB(K,BS,DB,DS,DD)
IF(K.EQ.1.AND.IBCINL.LT.0) CALL POLE(K)
IF(K.EQ.1.AND.IBCINL.LT.0) GO TO 999
IF(K.EQ.KMAX.AND.IBCFNL.LT.0) CALL POLE(K)
IF(K.EQ.KMAX.AND.IBCFNL.LT.0) GO TO 999
CALL PHIBET(K)
DEX=DECMX(K)
RRA=1./R(K)
OX=OXI(K)
OT=OMI(K)
GA=GAM(K)
DOXT=OX-OT
GDO=GA*DOXT
DD2D=DD2*DS
DO 3 MN=1,MNMAXO
EN=N(MN)
ENR=EN*RRA
CALL TLOAD(K)
TTS=TT(MN)*ALOAD
EX=(U3(MN)-U1(MN))*TDLI+OX*W2(MN)+ENL*OSE*(BX3(MN)+BE3(MN))
ET=ENR*V2(MN)+GA*U2(MN)+OT*W2(MN)+ENL*OSE*(BT3(MN)+BE3(MN))
EXT=J.5*((V3(MN)-V1(MN))*TDLI-ENR*U2(MN)-GA*V2(MN)
+ENL*SOE*BX3(MN))
1 KT=ENR*PHIT(MN)+GA*PHIX(MN)
KXT=J.5*((ENR*(-PHIX(MN)-GA*W2(MN)+(W3(MN)-W1(MN))*TDLI)+GDO*V2(MN)
+OT*(V3(MN)-V1(MN))*TDLI-GA*PHIT(MN)-DOXT*PHI(MN))
1 TX(MN)=BS*(EX+NU*EX)-TTS
TTH(MN)=BS*(ET+NU*EXT)
TXT(MN)=BS*D1*EXT
MK1=K1+(MN-1)*KMAX2
MX(MN)=Z(4,MK1)

```



```

MTH(MN)=NU*MX(MN)+DD2D*KT-D1*MT(MN)*ALOAD
MXT(MN)=DS*D1*KXT
MK11=MK1+1
MKK1=MK1-1
QS(MN)=SIGO*TKN*LAM2*(GA*MX(MN)+(Z(4,MK11)-Z(4,MKK1))*TDLI
1  +ENR*MXT(MN)-GA*MTH(MN)
MX(MN)=MX(MN)*ABZ3
MTH(MN)=MTH(MN)*ABZ3
MXT(MN)=MXT(MN)*ABZ3
TX(MN)=TX(MN)*ABZ
TTH(MN)=TTH(MN)*ABZ
TXI(MN)=TXI(MN)*ABZ
PHIX(MN)=PHIX(MN)*ABZO
PHIT(MN)=PHIT(MN)*ABZO
PHI(MN)=PHI(MN)*ABZO
U1(MN)=U2(MN)
U2(MN)=U3(MN)
V1(MN)=V2(MN)
V2(MN)=V3(MN)
W1(MN)=W2(MN)
W2(MN)=W3(MN)
FK=K-1
FIFREQ=IFREQ
KTST=(K-1)/IFREQ
FKTST=KTST
FKTEST=FK/FIFREQ-FKTST
IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 999
IF(FKTEST.NE.0.) GO TO 2
999 X(1,K)=0.
X(2,K)=0.
X(3,K)=0.
X(4,K)=0.
PTF(K)=0.
PF(K)=0.
AMX=0.
AMTH=0.
AMXTH=0.
ANX=0.
ANTH=0.
ANXTH=0.
AQS=0.
IF (JUMP.EQ.2) GO TO 73
DO 72 MN=1,MNMAXO
EN=N(MN)
FC=EN*THET
SN=SIN(FC)
CS=COS(FC)
X(1,K)=X(1,K)+U1(MN)*CS*ABZN
00001200
00001210
00001220
00001230
00001240
00001250
00001260
00001270
00001280
00001290
00001300
00001310
00001320
00001330
00001340
00001350
00001360
00001370
00001380
00001390
00001400
00001410
00001420
00001430
00001440
00001450
00001460
00001470
00001480
00001490
00001500
00001510
00001520
00001530
00001540
00001550
00001560
00001570
00001580
00001590
00001600
00001610
00001620
00001630
00001640
00001650
00001660
00001670

```



```

X(2,K)=X(2,K)+V1(MN)*SN*ABZN
X(3,K)=X(3,K)+W1(MN)*CS*ABZN
X(4,K)=X(4,K)+PHIX(MN)*CS
PTF(K)=PTF(K)+PHIT(MN)*SN
AMX=AMX+MX(MN)*CS
AMTH=AMTH+MTH(MN)*CS
AMXTH=AMXTH+MXTH(MN)*SN
ANX=ANX+TX(MN)*CS
ANTH=ANTH+TTH(MN)*CS
AQS=AQS+QS(MN)*CS
ANXTH=ANXTH+TXTH(MN)*SN
PF(K)=PF(K)+PHI(MN)*SN
GO TO 429
DO 74 MN=3,MNMAXO,JUMP
EN=N(MN)
FC=EN*THET
SN=SIN(FC)
CS=COS(FC)
MNM=MN-1
X(1,K)=X(1,K)+(U1(MN)*CS+V1(MNM)*SN)*ABZN
X(2,K)=X(2,K)+(V1(MN)*SN+V1(MNM)*CS)*ABZN
X(3,K)=X(3,K)+(W1(MN)*CS+W1(MNM)*SN)*ABZN
X(4,K)=X(4,K)+PHIX(MN)*CS+PHIX(MNM)*SN
PTF(K)=PTF(K)+PHIT(MN)*SN+PHIT(MNM)*CS
AMX=AMX+MX(MN)*CS+MX(MNM)*SN
AMTH=AMTH+MTH(MN)*CS+MTH(MNM)*SN
AMXTH=AMXTH+MXTH(MN)*SN+MXTH(MNM)*CS
ANX=ANX+TX(MN)*CS+TX(MNM)*SN
ANTH=ANTH+TTH(MN)*CS+TTH(MNM)*SN
ANXTH=ANXTH+TXTH(MN)*SN+TXTH(MNM)*CS
AQS=AQS+QS(MN)*CS+QS(MNM)*SN
PF(K)=PF(K)+PHI(MN)*SN+PHI(MNM)*CS
X(1,K)=X(1,K)+U1(1)*ABZN
X(2,K)=X(2,K)+V1(1)*ABZN
X(3,K)=X(3,K)+W1(1)*ABZN
X(4,K)=X(4,K)+PHIX(1)
PTF(K)=PTF(K)+PHIT(1)
PF(K)=PF(K)+PHI(1)
AMX=AMX+MX(1)
AMTH=AMTH+MTH(1)
AMXTH=AMXTH+MXTH(1)
ANX=ANX+TX(1)
ANTH=ANTH+TTH(1)
ANXTH=ANXTH+TXTH(1)
AQS=AQS+QS(1)
CONTINUE
IF(K.EQ.1) WRITE(6,117)
IF((K.EQ.1.AND.IBCINL.LT.0).OR.(K.EQ.KMAX.AND.IBCFNL.LT.0))

```

72

73

74

429


```

1 GO TO 220
  GO TO 221
220 WRITE (6,1181) K, ANX, ANTH, ANXTH, AMX, AMTH, AMXTH
221 GO TO 2
121 CONTINUE
  DO 660 K=1, KMAX
    FK=K-1
    FIFREQ=IFREQ
    KTST=(K-1)/IFREQ
    FKTEST=FK/FIFREQ-FKTST
    IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 661
    IF(FKTEST.NE.0.) GO TO 658
    IF(K.EQ.1) WRITE(6,217)
    WRITE(6,218) K, X(1,K), X(2,K), X(3,K), X(4,K), PTF(K), PF(K)
661 DO 659 I=1,4
658 X(I,K)=0.
659 CONTINUE
660 CONTINUE
991 IF(IMODE.LE.0) RETURN
  DO 534 MN=1, MNMAXO
    WRITE(6,749) N(MN)
  DO 521 MM=1, MNMAXO
    I1=1+(MM-1)*KMAX2
    I2=I1+1
    U1(MM)=Z(1,I1)
    U2(MM)=Z(1,I2)
    V1(MM)=Z(2,I1)
    V2(MM)=Z(2,I2)
    W1(MM)=Z(3,I1)
    W2(MM)=Z(3,I2)
  CONTINUE
521 DO 445 K=1, KMAX
    K1=K+1
    CALL BDB(K, BS, DB, DS, DD)
    IF(K.EQ.1.AND.IBCINL.LT.0) CALL POLE(K)
    IF(K.EQ.KMAX.AND.IBCFNL.LT.0) CALL POLE(K)
    TXZ=TX(MN)
    TTHZ=TTH(MN)
    TXTZ=TX(MN)
    AMXZ=MX(MN)
    AMTHZ=MTH(MN)
    AMXTZ=MX(MN)
    QSZ=QS(MN)
    X(1,K)=PHIX(MN)
    X(2,K)=PHIT(MN)

```



```

X(3,K)=PHI(MN)
IF(K.EQ.1.AND.IBCINL.LT.0) GO TO 583
IF(K.EQ.KMAX.AND.IBCFNL.LT.0) GO TO 583
CALL PHIBET(K)
DEX=DEOMX(K)
RRA=1./R(K)
OX=OMXI(K)
OT=OMT(K)
GA=GAM(K)
DOXT=OX-OT
GDO=GA*DOXT
DD2D=DD2*DS
EN=N(MN)
ENR=EN*RA
CALL TLOAD(K)
TTS=TT(MN)*ALOAD
EX=(U3(MN)-U1(MN))*TDLI+OX*W2(MN)+ENL*OSE*(BX3(MN)+BE3(MN))
ET=ENR*V2(MN)+GA*U2(MN)+OT*W2(MN)+ENL*OSE*(BT3(MN)+BE3(MN))
EXT=.5*((V3(MN)-V1(MN))*TDLI-ENR*U2(MN)-GA*V2(MN)
1 KT=ENR*PHIT(MN)+GA*PHIX(MN)
1 KXT=.5*(ENR*(-PHIX(MN)-GA*W2(MN)+(W3(MN)-W1(MN))*TDLI)+GDO*V2(MN)
+OT*((V3(MN)-V1(MN))*TDLI-GA*PHIT(MN)-DOXT*PHI(MN))
TXZ=(BS*(EX+NU*ET)-TTS)*ABZ
TTHZ=(BS*(ET+NU*EX)-TTS)*ABZ
TXYZ=BS*DI*EXT*ABZ
MK1=K1+(MN-1)*KMAX2
AMXZ=Z(4,MK1)
AMTHZ=NU*AMXZ+DD2D*KT-D1*MT(MN)*ALOAD
AMXTZ=DS*DI*KXT
MK11=MK1+1
MKK1=MK1-1
1 QSZ=SIGO*TKN*LAM2*(GA*AMXZ+(Z(4,MK11)-Z(4,MKK1))*TDLI
+ENR*AMXTZ-GA*AMTHZ)
AMXZ=AMXZ*ABZ3
AMTHZ=AMTHZ*ABZ3
AMXTZ=AMXTZ*ABZ3
X(1,K)=PHIX(MN)*ABZO
X(2,K)=PHIT(MN)*ABZO
X(3,K)=PHI(MN)*ABZO
DO 533 MM=1,MNMAXO
U1(MM)=U2(MM)
U2(MM)=U3(MM)
V1(MM)=V2(MM)
V2(MM)=V3(MM)
W1(MM)=W2(MM)
W2(MM)=W3(MM)
FK=K-1
533

```



```

FIFREQ=IFREQ
KTST=(K-1)/IFREQ
FKTST=KTST
FKTEST=FK/FIFREQ-FKTST
IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 583
IF(FKTEST.NE.0.) GO TO 445
583 CONTINUE
IF(K.EQ.1) WRITE(6,117)
IF((K.EQ.1.AND.IBCINL.LT.0).OR.(K.EQ.KMAX.AND.IBCFNL.LT.0))
1 GO TO 4444
GO TO 4445
4444 WRITE(6,1181) K, TXZ, TTHZ, TXTZ, AMXZ, AMTHZ, AMXTZ
GO TO 445
4445 WRITE(6,118) K, TXZ, TTHZ, TXTZ, QSZ, AMXZ, AMTHZ, AMXTZ
445 CONTINUE
WRITE(6,217)
DO 446 K=1,KMAX
FK=K-1
FIFREQ=IFREQ
KTST=(K-1)/IFREQ
FKTST=KTST
FKTEST=FK/FIFREQ-FKTST
IF(K.EQ.1.OR.K.EQ.KMAX) GO TO 593
IF(FKTEST.NE.0.) GO TO 446
593 KZ=K+1+(MN-1)*KMAX2
UP=Z(1,KZ)*ABZN
VP=Z(2,KZ)*ABZN
WP=Z(3,KZ)*ABZN
WRITE(6,218) K, UP, VP, WP, X(1,K), X(2,K), X(3,K)
446 CONTINUE
534 CONTINUE
C *****
101 FORMAT(1H1,34H
1E LOAD FACTOR IS E11.4,36H
21H ITERATIONS////)
102 FORMAT(1H0,35H
103 FORMAT(//7H N 16H PHI S 16H U 16H PHI THETA10H
1W 16H PHI S 16H U 16H PHI THETA10H
104 FORMAT(2X,12,3X,6E16.4)
105 FORMAT(7H N 16H Q S
1STHETA 16H M STHETA /)
2H
106 FORMAT(1H0,8HSTATION I3,14H- THETA OUTPUT / )
107 FORMAT(
14H THETA 14H U 14H PHI THETA14H
1V PHI
2
108 FORMAT(8E14.4)
109 FORMAT(1H0,
14H THETA 14H M S 14H

```



```

1  M THETA 14H M STHETA 14H N S 14H N THETA 14H 1400003600
2  N STHETA 14H Q S 00003610
114 FORMAT(2X,I2,3X,7E16.4) 00003620
115 FORMAT(7H N 16H M S 16H THETA 15H M STHETA//) 00003630
116 FORMAT(1H0,84H THE SUMMED FORCES, MOMENTS, DISPLACEMENTS 00003640
117 AND ROTATIONS FOLLOW FOR THETA=E15.6//) 00003650
117 IN STHETA 16H N S 16H THETA 16H M THETA 00003660
216H M STHETA //) 00003670
118 FORMAT(1X,I3,3X,7E16.4) 00003680
1181 FORMAT(1X,I3,3X,3E16.4) NOT COMPUTED '3E16.4' 00003690
151 FORMAT(1H1,26H THE TIME STEP THE SOLUTION CONVERGED IN 12,11H IT 00003700
1 2,4H OR E9.3,40H SECONDS 00003710
3ERATIONS////) 00003720
217 FORMAT(//8H STATION,15H U 16H V 16H PHI 00003730
1 W 16H PHI THETA16H 00003740
2 /) 00003750
218 FORMAT(1X,I3,3X,6E16.4) 00003760
749 FORMAT(1H1,40X,27H MODAL OUTPUT FOR MODE N = 13,8H FOLLOWS) 00003770
RETURN 00003780
END 00003790
00003800
00003810
00003820

```

```

SUBROUTINE IMPERF
COMMON /IBL11/ ICORFL,IPASS,JUMP,MPERFS
COMMON /BLIMP/ PHIXB(951),PHITB(951)
DIMENSION FNT(25),DFNT(25)
** THIS SUBROUTINE EITHER CAN COMPUTE THE INITIAL IMPERFECTIONS OF **
** THE SHELL AND INITIALIZE ARRAYS PHIXB AND PHITB, OR JUST CAUSE **
** A LOADING OF THE ARRAYS FROM PRE-PREPARED DATA CARDS. IF 'MPERFS' **
** IS 0, THEN ARRAYS PHIXB AND PHITB ARE MADE ZERO. THE IMPERFECTION **
** NOTE: WHEN LOADING ARRAYS PHIXB AND PHITB WITH ORDERING MUST BE **
** NOTATION FOURIER COEFFICIENTS, THE FOLLOWING MODE '0', **
** FIRST KMAX LOCATIONS CONTAIN MODE '0', **
** SECOND KMAX LOCATIONS CONTAIN MODE '1', **

```


APPENDIX G

LISTING OF IMPERFECTION DATA HANDLING PROGRAM

```

C      PROGRAM TO PREPARE RAW IMPERFECTION DATA FOR USE BY SATANS-II
DIMENSION A14(1422),AM14(35,49),PHIXRO(35,49),PHITRO(35,49),
1      TEMP(48),A(25),B(25),PXROCC(35,25),PXROSC(35,25),
2      PTRJCC(35,25),PTRJSC(35,25),PXRCCS(25),PXRSCS(25),
3      PTRCCS(25),PTRSCS(25),CHECK(361),C(25),S(25),NN(13),
4      PHIXB(951),PHITB(951),E(25)
00000010
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
00000100
00000110
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420

C      READ RAW DATA INTO COLUMN MATRIX, A14:
DO 1 I=1,237
K=6*(I-1)
1 READ (5,1000) A14(K+1),A14(K+2),A14(K+3),A14(K+4),A14(K+5),
A14(K+6)
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420

C      TRANSFER DATA INTO 35X49 MATRIX: (NOTE - ROWS 1,2,3,33,34,35 ARE
UNKNOWN AT THIS POINT)
IN THE PROCESS, DIMENSIONALIZE THE DATA (MULT. BY SHELL THICKNESS)
K=0
DO 2 I=4,32
DO 2 J=1,49
K=K+1
2 AM14(I,J)=A14(K)*0.00437
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420

C      PERFORM MODIFIED LINEAR EXTRAPOLATION FOR END POINTS
DO 3 J=1,49
D1=(AM14(4,J)+AM14(5,J))/2.0
D2=(AM14(6,J)+AM14(7,J))/2.0
D3=(AM14(32,J)+AM14(31,J))/2.0
D4=(AM14(30,J)+AM14(29,J))/2.0
THETAB=(D1-D2)/0.48
THETAB=(D3-D4)/0.48
AM14(3,J)=D1+THETAB*0.36
AM14(2,J)=D1+THETAB*0.6
AM14(1,J)=D1+THETAB*0.84
AM14(33,J)=D3+THETAB*0.36
AM14(34,J)=D3+THETAB*0.6
3 AM14(35,J)=D3+THETAB*0.84
00000120
00000130
00000140
00000150
00000160
00000170
00000180
00000190
00000200
00000210
00000220
00000230
00000240
00000250
00000260
00000270
00000280
00000290
00000300
00000310
00000320
00000330
00000340
00000350
00000360
00000370
00000380
00000390
00000400
00000410
00000420

C      FULL 35X49 MATRIX OF DIMENSIONAL DEVIATIONS EXISTS.

```



```

WRITE (6,1001)
WRITE (6,1002)
WRITE (6,1003) ((AM14(I,J),J=1,49),I=1,35)
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900

COMPUTE MERIDIONAL ROTATIONS USING FINITE DIFFERENCING

DO 4 J=1,49
DO 5 I=2,34
IM1=I-1
IP1=I+1
5 PHIXRO(I,J)=(AM14(IM1,J)-AM14(IP1,J))/0.48
4 PHIXRO(35,J)=PHIXRO(34,J)
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900

COMPUTE CIRCUMFERENTIAL ROTATIONS IN SIMILAR FASHION
NOTE: SPACING EQUALS CIRCUMFERENCE/INCREMENTS = PI/6)

DO 6 I=1,35
DO 7 J=2,48
JM1=J-1
JP1=J+1
7 PHITRO(I,J)=(AM14(I,JM1)-AM14(I,JP1))/1.047198
6 PHITRO(I,1)=(AM14(I,48)-AM14(I,2))/1.047198
6 PHITRO(I,49)=PHITRO(I,1)
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900

FULL 35X49 MATRICES OF MERIDIONAL AND CIRCUMFERENTIAL ROTATIONS
EXIST.

WRITE (6,1001)
WRITE (6,1004)
WRITE (6,1003) ((PHIXRO(I,J),J=1,49),I=1,35)
WRITE (6,1001)
WRITE (6,1003) ((PHITRO(I,J),J=1,49),I=1,35)
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900

NONDIMENSIONALIZE THE PHIXRO AND PHITRO MATRICES FOR USE BY
SATANS-II - MULTIPLY BY EO/SIGO

DO 77 I=1,35
DO 77 J=1,49
PHIXRO(I,J)=PHIXRO(I,J)*0.158E5
PHITRO(I,J)=PHITRO(I,J)*0.158E5
77 WRITE (6,1001)
WRITE (6,2004) ((PHIXRO(I,J),J=1,49),I=1,35)
WRITE (6,1001)
WRITE (6,2005) ((PHITRO(I,J),J=1,49),I=1,35)
00000430
00000440
00000450
00000460
00000470
00000480
00000490
00000500
00000510
00000520
00000530
00000540
00000550
00000560
00000570
00000580
00000590
00000600
00000610
00000620
00000630
00000640
00000650
00000660
00000670
00000680
00000690
00000700
00000710
00000720
00000730
00000740
00000750
00000760
00000770
00000780
00000790
00000800
00000810
00000820
00000830
00000840
00000850
00000860
00000870
00000880
00000890
00000900

```



```
C
C
C      NOW EXPAND EACH CIRCUMFERENCE OF PHIXRO'S AND PHITRO'S INTO 49
C      ELEMENT FOURIER SERIES. PUT SINE AND COSINE COEFFS. INTO ARRAYS
C      PXROCC,PXROSC,PTROCC,PTROSC.
C
DO 8 I=1,35
DO 9 J=1,48
TEMP(J)=PHIXRJ(I,J)
CALL FORIER(48,TEMP,25,AH0,SYS,A,B,E)
DO 901 KK=1,24
K=25-KK
KPI=K+I
A(KPI)=A(K)
B(KPI)=B(K)
A(I)=AHO
B(I)=O.O
DO 10 J=1,25
PXROCC(I,J)=A(J)
PXROSC(I,J)=B(J)
DO 11 J=1,48
TEMP(J)=PHITRJ(I,J)
CALL FORIER(48,TEMP,25,AH0,SYS,A,B,E)
DO 902 KK=1,24
K=25-KK
KPI=K+I
A(KPI)=A(K)
B(KPI)=B(K)
A(I)=AHO
B(I)=O.O
DO 12 J=1,25
PTROCC(I,J)=A(J)
PTROSC(I,J)=B(J)
8 CONTINUE
C
C      A 49-ELEMENT FOURIER EXPANSION FOR MERIDIONAL AND CIRCUMFERENTIAL
C      ROTATION IMPERFECTIONS EXISTS FOR EACH CIRCUMFERENCE.
C
WRITE(6,1001)
WRITE(6,1006)
DO 13 I=1,35
WRITE(6,1007) (PXROSC(I,J),J=1,25),(PTROSC(I,J),J=1,25)
WRITE(6,1001)
WRITE(6,1008)
DO 14 I=1,35
WRITE(6,1007) (PTROCC(I,J),J=1,25),(PTROSC(I,J),J=1,25)
C
C      DETERMINE RELATIVE IMPORTANCE OF EACH MODAL COMPONENT OF THE MERI-
C      DIONAL AND CIRCUMFERENTIAL ROTATIONS
C
```


C

```

DO 15 J=1,25
PXCSUM=0.0
PXSSUM=0.0
PTCSSUM=0.0
PTSSUM=0.0
DO 16 I=1,35
PXCSUM=PXCSUM+ABS(PXROCC(I,J))
PXSSUM=PXSSUM+ABS(PXROSC(I,J))
PTCSSUM=PTCSSUM+ABS(PTROCC(I,J))
PTSSUM=PTSSUM+ABS(PTROSC(I,J))
16 PXRCCS(J)=PXCSUM/35.0
PXRSCS(J)=PXSSUM/35.0
PTRCCS(J)=PTCSSUM/35.0
PTRSCS(J)=PTSSUM/35.0
15 WRITE(6,1001)
WRITE(6,1009)
DO 17 I=1,25
J=I-1
17 WRITE(6,1010) J,PXRCCS(I),PXRSCS(I),PTRCCS(I),PTRSCS(I)

```

C
C
C
C

```

PICK CIRCUMFERENCE 10 AND COMPARE ACTUAL VS FOURIER REPRESENTATION
OF CIRCUMFERENTIAL ROTATION IMPERFECTION.

```

```

DO 18 I=1,25
A(I)=PTROCC(10,I)
B(I)=PTROSC(10,I)
18 STEP=.1745329E-1
DO 19 I=1,361
THETA=FLOAT(I-1)*STEP
DO 20 J=2,25
RJ=FLOAT(J-1)
C(J)=A(J)*COS(RJ*THETA)
S(J)=B(J)*SIN(RJ*THETA)
20 CHECK(I)=A(I)
DO 21 J=2,25
CHECK(I)=CHECK(I)+C(J)+S(J)
21 CONTINUE
19 WRITE(6,1001)
WRITE(6,1011)
K=1
DO 22 I=1,361,15
II=I-1
22 WRITE(6,1012) II,PHITRO(10,K),CHECK(I)
K=K+2
DO 23 I=1,90
II=I-1

```


C
C
C
FIFTH KMAX LOCNS ARE MODE 'J',
SIXTH KMAX LOCNS ARE MODE '-K', ETC.....

101	DO 101 I=1,35 PHIXB(I)=PXROCC(I,1) PHITB(I)=PTROCC(I,1) DO 102 I=1,35 II=I+70 JJ=I-35 PHIXB(II)=PXROCC(I, NN(2)) PHITB(II)=PTROCC(I, NN(2)) PHIXB(JJ)=PXROSC(I, NN(2)) PHITB(JJ)=PTROSC(I, NN(2)) DO 103 I=1,35 II=I+140 JJ=I-35 PHIXB(II)=PXROCC(I, NN(3)) PHITB(II)=PTROCC(I, NN(3)) PHIXB(JJ)=PXROSC(I, NN(3)) PHITB(JJ)=PTROSC(I, NN(3)) DO 104 I=1,35 II=I+210 JJ=I-35 PHIXB(II)=PXROCC(I, NN(4)) PHITB(II)=PTROCC(I, NN(4)) PHIXB(JJ)=PXROSC(I, NN(4)) PHITB(JJ)=PTROSC(I, NN(4)) DO 105 I=1,35 II=I+280 JJ=I-35 PHIXB(II)=PXROCC(I, NN(5)) PHITB(II)=PTROCC(I, NN(5)) PHIXB(JJ)=PXROSC(I, NN(5)) PHITB(JJ)=PTROSC(I, NN(5)) DO 106 I=1,35 II=I+350 JJ=I-35 PHIXB(II)=PXROCC(I, NN(6)) PHITB(II)=PTROCC(I, NN(6)) PHIXB(JJ)=PXROSC(I, NN(6)) PHITB(JJ)=PTROSC(I, NN(6)) DO 107 I=1,35 II=I+420 JJ=I-35 PHIXB(II)=PXROCC(I, NN(7)) PHITB(II)=PTROCC(I, NN(7)) PHIXB(JJ)=PXROSC(I, NN(7)) PHITB(JJ)=PTROSC(I, NN(7))	00002350 00002360 00002370 00002380 00002390 00002400 00002410 00002420 00002430 00002440 00002450 00002460 00002470 00002480 00002490 00002500 00002510 00002520 00002530 00002540 00002550 00002560 00002570 00002580 00002590 00002600 00002610 00002620 00002630 00002640 00002650 00002660 00002670 00002680 00002690 00002700 00002710 00002720 00002730 00002740 00002750 00002760 00002770 00002780 00002790 00002800 00002810 00002820
102		
103		
104		
105		
106		
107		

DO 108	I=1,35		00002830
II=I+490			00002840
JJ=I-35			00002850
PHIXB(II)=PXROCC(I, NN(8))			00002860
PHITB(II)=PTRJCC(I, NN(8))			00002870
PHIXB(JJ)=PXROSC(I, NN(8))			00002880
PHITB(JJ)=PTRJSC(I, NN(8))			00002890
DO 109	I=1,35		00002900
II=I+560			00002910
JJ=I-35			00002920
PHIXB(II)=PXROCC(I, NN(9))			00002930
PHITB(II)=PTRJCC(I, NN(9))			00002940
PHIXB(JJ)=PXROSC(I, NN(9))			00002950
PHITB(JJ)=PTRJSC(I, NN(9))			00002960
DO 110	I=1,35		00002970
II=I+630			00002980
JJ=I-35			00002990
PHIXB(II)=PXROCC(I, NN(10))			00003000
PHITB(II)=PTRJCC(I, NN(10))			00003010
PHIXB(JJ)=PXROSC(I, NN(10))			00003020
PHITB(JJ)=PTRJSC(I, NN(10))			00003030
DO 111	I=1,35		00003040
II=I+700			00003050
JJ=I-35			00003060
PHIXB(II)=PXROCC(I, NN(11))			00003070
PHITB(II)=PTRJCC(I, NN(11))			00003080
PHIXB(JJ)=PXROSC(I, NN(11))			00003090
PHITB(JJ)=PTRJSC(I, NN(11))			00003100
DO 112	I=1,35		00003110
II=I+770			00003120
JJ=I-35			00003130
PHIXB(II)=PXROCC(I, NN(12))			00003140
PHITB(II)=PTRJCC(I, NN(12))			00003150
PHIXB(JJ)=PXROSC(I, NN(12))			00003160
PHITB(JJ)=PTRJSC(I, NN(12))			00003170
DO 113	I=1,35		00003180
II=I+840			00003190
JJ=I-35			00003200
PHIXB(II)=PXROCC(I, NN(13))			00003210
PHITB(II)=PTRJCC(I, NN(13))			00003220
PHIXB(JJ)=PXROSC(I, NN(13))			00003230
PHITB(JJ)=PTRJSC(I, NN(13))			00003240
WRITE (6,1001)			00003250
WRITE (6,1016)			00003260
WRITE (6,1017)		(PHIXB(I), I=1,875)	00003270
WRITE (6,1001)			00003280
WRITE (6,1018)			00003290
WRITE (6,1017)		(PHITB(I), I=1,875)	00003300


```

SUBROUTINE FORIER (LP,Y,M,AH0,SYS,A,B,E)
GIVEN A SET OF LP DATA POINTS Y(Q), EQUALLY SPACED, FORIER WILL DETER
MINE LP CONSTANTS SO THAT EITHER EQUATION B.1 OR B.2 BELOW IS
SATISFIED:
    IF LP BE AN ODD NUMBER, THEN N IS TAKEN AS (L-1)/2, AND;
B.1:
    P=1/_
    Y(N)=A(0)/2 + SUM/ A(P)COS(2PI*NP/(2N+1)) + B(P)SIN(2PI*NP/(2N+1)) /
    WHERE N=0,1,2,...,2N
    IF LP BE AN EVEN NUMBER, THEN N IS TAKEN AS L/2 AND;
B.2:
    N-1/_
    Y(N)=A(0)/2+SUM/ (A(P)COS(PI*NP/N)+B(P)SIN(PI*NP/N))+A(N)COS(PI*N))/2/
    WHERE N=0,1,2,...,2N-1
CONTINUE
ARGUMENTS:
LP:  NUMBER OF DATA POINTS TO BE FITTED, AN INTEGER. THIS
    IS THE NJMBER OF ITEMS IN THE Y ARRAY TO BE CONSIDERED
    BY FORIER. (DO NOT INCLUDE THE POINT AT 2PI)
Y:   DATA ARRAY TO BE DECLARED IN USER'S DIMENSION STATEMENT
    AS A REAL*4. IT MUST CONTAIN AT LEAST LP MEANGFUL
    ITEMS. Y DATA ARE ALWAYS TAKEN TO BE EQUALLY SPACED.
M:   HIGHEST HARMONIC TO BE CONSIDERED IN MAKING THE FIT, AN
    INTEGER. IF M<=0 OR M>N, THEN THE EFFECTIVE M USED BY
    FORIER WILL BE TAKEN AS EQUAL TO N.
AH0: THE RESULT A(0)/2.
SYS: THE SUM OF Y*Y.
A:   AN ARRAY DECLARED IN USER'S DIMENSION STATEMENT. A(1)
    CONTAINS THE FOURIER COEFFICIENT A1, A(2) CONTAINS A2, ETC.
    NOTE: A(N) CONTAINS A(N/2) WHEN LP IS AN EVEN NUMBER.
    NOTE: A MUST BE DIMENSIONED AT LEAST N IN SIZE.
B:   SIMILAR TO THE A ARRAY. CONTAINS THE SINE COEFFICIENTS.
    NOTE: WHEN LP IS EVEN, B(N) IS SET TO 0.0.
E:   AN ARRAY DECLARED IN USER'S DIMENSION STATEMENT. E(I)
    CONTAINS THE SUM OF THE SQUARES OF THE ERRORS IN THE FIT
    WHEN TERMS ONLY UP TO I, HARMONICS ARE USED. IF N HARMONICS
    ARE USED, E(N) IS ZERO BY DEFINITION BUT IS CALCULATED
    ANALYTICALLY TO PROVIDE SOME CONTROL ON ACCURACY OF CALC.
DIMENSION Y(1),A(1),B(1),E(1)
L=LP

```

CC


```

28  Q=C+Q
    IM=IM+IL
    IP=N
    GO TO 255
253  A(N)=AAA/2.0
    B(N)=0.0
    SUMC=SUMC+A(N)*AAA
    GO TO 254
33  RETURN
    END

```

```

00004730
00004740
00004750
00004760
00004770
00004780
00004790
00004800
00004810
00004820

```


LIST OF REFERENCES

1. Bushnell, D., Computer Analysis of Shell Structures, paper 69-WA/PVP-13 presented at the ASME Winter Annual Meeting, Los Angeles, California, November 1969.
2. Air Force Flight Dynamics Laboratory Report AFFDL-TR-71-54, An Assessment of Current Capability for Computer Analysis of Shell Structures, by R. F. Hartung, April 1971.
3. National Aeronautics and Space Administration Report NASA-CR-909, A Geometrically Nonlinear Analysis of Arbitrarily Loaded Shells of Revolution, by R. E. Ball, January 1968.
4. National Aeronautics and Space Administration Report NASA-CR-1987, A Computer Program for the Geometric Nonlinear Static and Dynamic Analysis of Arbitrarily Loaded Shells of Revolution, Theory and User's Manual, by R. E. Ball, April 1972.
5. Dantone, J. J., A Computer Study of the Buckling of an Imperfect Cylindrical Shell, M.S. Thesis, Department of Aeronautics, Naval Postgraduate School, Monterey, California, September 1971.
6. National Aeronautics and Space Administration Report NASA-CR-1163, Experimental Investigation of the Effect of General Imperfections on the Buckling of Cylindrical Shells, by J. Arbocz and C. D. Babcock, Jr., September 1968.
7. Sanders, J. Jr., "Nonlinear Theories for Thin Shells," Quarterly Journal of Applied Mathematics, v. 21, p. 21-36, 1963.
8. National Advisory Committee on Aeronautics Report 1010, A Recurrence Matrix Solution for the Dynamic Response of Aircraft in Gusts, by J. C. Houbolt, 1951.
9. Mathematics Centrum, Amsterdam, Holland Report MR19, A Matrix Method for the Solution of a Second Order Difference Equation in Two Variables, by M. L. Potters, 1955.

10. Air Force Flight Dynamics Laboratory/Lockheed Missiles and Space Company Report AFFDL-TR-71-79, Computer Oriented Analysis of Shell Structures, "Design Problems of Shell Structures and the Impact of the Computer on Shell Analysis," by M. Stein, p. 14, June 1971.
11. W. R. Church Computer Center User's Manual, Appendix C, Naval Postgraduate School, Monterey, California, 1971.
12. Texas A&M University Aerospace Engineering Department Report 69-77, Nonlinear Dynamic Analysis of Shells of Revolution by Matrix Displacement Method, by J. A. Stricklin, and others, February 1970.
13. The Catholic University of America, A Note on Buckling of Spherical Caps with Initial Asymmetric Imperfections, by Robert Kao, December 1971.
14. National Aeronautics and Space Administration Report NASA-CR-161, Influence of Edge Conditions on the Stability of Axially Compressed Cylindrical Shells, by B. O. Almroth, February 1965.
15. National Aeronautics and Space Administration Report NASA-CR-1998, A Digital Computed Study of the Buckling of Shallow Spherical Caps and Truncated Hemispheres, by W. C. Stilwell and R. E. Ball, June 1972.
16. Bushnell, D., Almroth, B. O., and Brogan, F.A., "Finite Difference Energy Method for Nonlinear Shell Analysis," Journal of Computers and Structures, v. 1, p.361-387, 1971.
17. Huang, N. C., "Unsymmetrical Buckling of Thin Shallow Shells," Journal of Applied Mechanics, v. 31, p. 447-457, September 1964.
18. Famili, J., and Archer, R. R., "Finite Asymmetric Deformation of Shallow Spherical Shells," Journal of the American Institute of Aeronautics and Astronautics, v. 3, p. 506-510, March 1965.
19. American Institute of Aeronautics and Astronautics Paper No. 68-292, Nonlinear Deflection of Asymmetrically Loaded Shells of Revolution, by H. G. Schaeffer and R. E. Ball, April 1968.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Professor Robert E. Ball Department of Aeronautics Naval Postgraduate School Monterey, California 93940	2
4. Chairman, Department of Aeronautics Naval Postgraduate School Monterey, California 93940	1
5. LCDR Bruce A. Ryan, USN 148 Lincoln Place Waldwick, New Jersey 07463	1
6. Professor Johann Arbocz Firestone Flight Sciences Laboratory California Institute of Technology Pasadena, California 91109	1
7. Professor Charles Babcock Firestone Flight Sciences Laboratory California Institute of Technology Pasadena, California 91109	1
8. Dr. Richard F. Hartung, Manager Structural Mechanics Laboratory Lockheed - Palo Alto Research Laboratories 3251 Hanover Street Palo Alto, California 94304	1
9. Dr. Bö O. Almroth Structural Mechanics Laboratory Lockheed - Palo Alto Research Laboratories 3251 Hanover Street Palo Alto, California 94304	1
10. Dr. Frank A. Brogan Structural Mechanics Laboratory Lockheed - Palo Alto Research Laboratories 3251 Hanover Street Palo Alto, California 94304	1

11. Dr. David Bushnell 1
Structural Mechanics Laboratory
Lockheed - Palo Alto Research Laboratories
3251 Hanover Street
Palo Alto, California 94304
12. Professor Lars Åke Samuelson 1
The Aeronautical Research Institute of Sweden
P. O. Box 11021
S-161 11 Bromma 11
SWEDEN
13. Mr. Richard Citerly 1
Anamet Laboratories
P. O. Box 831
San Carlos, California 94070
14. Dr. Robert Fulton 1
Structural Mechanics Branch
NASA - Langley Research Center
Hampton, Virginia 23365
15. Mr. John A. O'Malley 1
Code 45
Naval Weapons Center
China Lake, California 93555

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

ORIGINATING ACTIVITY (Corporate author)

Naval Postgraduate School
Monterey, California 93940

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

REPORT TITLE

A Digital Computer Study of the Buckling of Actual Imperfect
Cylinders - A Modification of the Computer Program SATANS - Theory and
User's Manual for SATANS-I and SATANS-II.

DESCRIPTIVE NOTES (Type of report and inclusive dates)

Thesis for the degree of Aeronautical Engineer, December 1972.

AUTHOR(S) (First name, middle initial, last name)

Bruce A. Ryan, LCDR, USN

REPORT DATE

December 1972

7a. TOTAL NO. OF PAGES

268

7b. NO. OF REFS

19

CONTRACT OR GRANT NO.

9a. ORIGINATOR'S REPORT NUMBER(S)

PROJECT NO.

9b. OTHER REPORT NO(S) (Any other numbers that may be assigned
this report)

DISTRIBUTION STATEMENT

Approved for public release; distribution unlimited.

SUPPLEMENTARY NOTES

Work was partially supported by
Code 45, Naval Weapons Center,
China Lake, California 93555.

12. SPONSORING MILITARY ACTIVITY

Naval Postgraduate School
Monterey, California 93940

ABSTRACT A digital computer program for the geometrically nonlinear static and dynamic response of arbitrarily loaded shells of revolution known as SATANS is modified, removing a requirement that the applied loads be symmetric about a datum meridional plane. The ability to include arbitrary naturally-occurring initial geometric imperfections of any shape, location or extent is added to the program. Two specific axially loaded circular cylinders are analyzed using actual initial imperfections. Detailed numerical studies are conducted to determine the buckling loads of the perfect and imperfect cylinders. Several sets of boundary conditions are considered. An analysis procedure for imperfect shells is proposed. The results indicate that a means to conduct a numerical analysis of an arbitrarily imperfect shell of revolution has been developed.

A study of the buckling behavior of the clamped shallow spherical cap modelled under various boundary conditions at the pole is included as an addendum.

FORM 1473

(PAGE 1)

Unclassified

Security Classification

A-31408

KEY WORDS

LINK A

LINK B

LINK C

ROLE

WT

ROLE

WT

ROLE

WT

SHELLS

SHELLS OF REVOLUTION

STRUCTURAL ANALYSIS

BUCKLING

BUCKLING OF SHELLS

BUCKLING OF IMPERFECT SHELLS

IMPERFECTIONS

IMPERFECT SHELLS

Thesis

R94

c.1

Ryan

139074

A digital computer study of the buckling of actual imperfect cylinders - a modification of the computer program SATANS - theory and user's model for SATANS-I and SATANS-II.

Thesis

R94

c.1

Ryan

139074

A digital computer study of the buckling of actual imperfect cylinders - a modification of the computer program SATANS - theory and user's model for SATANS-I and SATANS-II.

thesR94

A digital computer study of the buckling



3 2768 001 97024 7

DUDLEY KNOX LIBRARY